Métaheuristiques pour l'optimisation multiobjectif: Approches coopératives, prise en compte de l'incertitude et application en logistique

Arnaud Liefooghe

Université Lille 1 - Sciences et Technologies
École Doctorale Sciences Pour l'Ingénieur Université Lille Nord-de-France
Laboratoire d'Informatique Fondamentale de Lille (UMR CNRS 8022)
Centre de Recherche INRIA Lille - Nord Europe

Mémoire de thèse présenté pour obtenir le grade de
Docteur de l'Université Lille 1 - Sciences et Technologies

Discipline : Informatique

Métaheuristiques pour l'optimisation multiobjectif

Approches coopératives, prise en compte de l'incertitude et application en logistique

---

Arnaud Liefooghe

Date de soutenance : 08/12/2009

Membres du jury

Président : Nour-Eddine Oussous, Professeur des Universités, Lille 1

Rapporteur : Alexandre Caminada, Professeur des Universités, Belfort-Montbéliard

Rapporteur : Jin-Kao Hao, Professeur des Universités, Angers

Examinateur : Dirk Thierens, Professeur des Universités, Utrecht (Pays-Bas)

Examinateur : Daniel Tuyttens, Professeur des Universités, Mons (Belgique)

Directeur : El-Ghazali Talbi, Professeur des Universités, Lille 1

Co-directeur : Laetitia Jourdan, Chargé de Recherche, INRIA Lille-Nord Europe

Numéro d'ordre : 40153 | Année : 2009

Résumé

De nombreux problèmes d'optimisation issus du monde réel, notamment dans le domaine de la logistique, doivent faire face à beaucoup de difficultés. En effet, ils sont souvent caractérisés par des espaces de recherche vastes et complexes, de multiples fonctions objectif contradictoires, et une foule d'incertitudes qui doivent être prises en compte. Les métaheuristiques sont des candidates naturelles pour résoudre ces problèmes, ce qui les rend préférables aux méthodes d'optimisation classiques. Toutefois, le développement de métaheuristiques efficaces découle d'un processus de recherche complexe. Le cœur de ce travail réside en la conception, l'implémentation et l'analyse expérimentale de métaheuristiques pour l'optimisation multiobjectif, ainsi que leurs applications à des problèmes logistiques de tournées et d'ordonnancement. Tout d'abord, une vue unifiée de ces approches est présentée, puis intégrée dans une plateforme logicielle dédiée à leur implémentation, ParadisEO-MOEO. Ensuite, plusieurs approches de coopération, combinant des métaheuristiques pour l'optimisation multiobjectif, sont proposées. Enfin, la question de la prise en compte l'incertitude est abordée dans le contexte de l'optimisation multiobjectif.

Mots clés. Recherche opérationnelle ; Optimisation combinatoire ; Optimisation multiobjectif ; Programmation heuristique ; Algorithmes hybrides ; Incertitude ; Ordonnancement ; Problème du voyageur de commerce.

Abstract

Many real-world optimization problems, especially in the field of logistics, have to face a lot of difficulties. Indeed, they are often characterized by large and complex search spaces, multiple conflicting objective functions, and a host of uncertainties that have to be taken into account. Metaheuristics are natural candidates to solve those problems and make them preferable to classical optimization methods. However, the development of efficient metaheuristics results in a complex engineering process. The core subject of this work lies in the design, implementation and experimental analysis of metaheuristics for multiobjective optimization, together with their applications to logistic problems from routing and scheduling. Firstly, a unified view of such approaches is presented and then integrated into a software framework for their implementation, ParadisEO-MOEO. Next, some cooperative approaches combining metaheuristics for multiobjective optimization are proposed. At last, the issue of uncertainty handling is discussed in the context of multiobjective optimization.

Keywords. Operational research ; Combinatorial optimization ; Multiobjective optimisation ; Heuristic programming ; Hybrid algorithms ; Uncertainty ; Scheduling ; Travelling Salesman Problem.

Remerciements

Mes premiers remerciements s'adressent à mes directeurs de thèse, El-Ghazali Talbi et Laetitia Jourdan. Je tiens à leur exprimer ma profonde reconnaissance pour leurs précieux conseils, la qualité de leur encadrement et leur lucidité scientifique, qui ont été primordiaux tout au long de ces années de doctorat.

Merci également à tous les membres du jury : Nour-Eddine Oussous, président, Alexandre Caminada et Jin-Kao Hao, rapporteurs, ainsi que Dirk Thierens et Daniel Tuyttens, examinateurs de cette thèse. À tous, je leur exprime ma sincère gratitude pour toute l'attention qu'ils ont porté à mon travail.

Je tiens par ailleurs à remercier toutes les personnes avec qui j'ai pris beaucoup de plaisirs à travailler durant ces trois années de thèse, tout autant les membres du LIFL, de l'INRIA et de l'équipe Dolphin que les collaborateurs extérieurs. Un merci particulier à Matthieu Basseur, Clarisse Dhaenens, Nicolas Jozefowiez, Nouredine Melab, mais également à José R. Figueira, Andrzej P. Wierzbicki et Edmund K. Burke. Merci également à tous les doctorants, post-doctorants, ingénieurs et stagiaires de l'équipe Dolphin, présents et passés. Je pense à Alex, Ali, Clive, Emilia, Gaël, Jean-Charles, Jérôme, Julien, Malika, Marie-Éléonore (merci pour les relectures), Mohamed, Mohand, Rémy, Salma, Thé-Van, Thomas, Waldo, et j'en oublies ! Un immense merci tout à fait particulier à Jérémie Humeau. Je suis conscient que cette thèse ne serait pas ce qu'elle est sans lui.

J'exprime enfin mes remerciements à ma famille et à mes amis, qui, par leur bonne humeur, sympathie et amitié ont contribué au bon déroulement de cette thèse. Merci à mes parents pour leur soutien infaillible tout au long de mon doctorat, mais aussi de mes études, et de mon existence en générale. Merci à ma soeur, Aude, de m'avoir ouvert la voie. Merci à ma femme, Anne, pour sa patience et son soutien. Merci enfin à mes fille, Pia et Cléo. À tous, ce manuscrit leur est dédié.

Table des matières

Introduction générale

Cette thèse s'inscrit dans le domaine de l'optimisation multiobjectif, et en particulier de l'étude et de la résolution de problèmes d'optimisation combinatoires et difficiles issus de la logistique. Elle a été menée au sein de l'équipe DOLPHIN [1] du laboratoire d'informatique fondamentale de Lille (LIFL, CNRS, Université Lille 1), commune à l'équipe-projet du même nom du centre de recherche INRIA Lille-Nord Europe.

Beaucoup de problèmes d'optimisation issus du monde réel sont complexes et difficiles à résoudre. De plus, les problèmes d'optimisation rencontrés en pratique sont rarement monoobjectif, et requièrent souvent la prise en compte de plusieurs critères conflictuels, ce qui amplifie encore le degré de difficulté. Ainsi, de nombreux domaines industriels, notamment en logistique, se doivent de faire face à des problèmes multiobjectif complexes et de grande taille. D'autant que la complexité des problèmes d'optimisation multiobjectif augmente en fonction de la taille du problème à résoudre, aussi bien en termes d'espace de recherche qu'en termes de nombre de fonctions objectif à optimiser. Aussi, bien des problèmes multiobjectif académiques et industriels sont NP-difficiles, et ne peuvent donc généralement pas être résolus de façon exacte en un temps de calcul raisonnable. Il s'avère donc nécessaire de considérer des méthodes alternatives. Pour cela, les métaheuristiques ont connu un intérêt grandissant au cours des deux dernières décennies. De nos jours, les métaheuristiques pour l'optimisation multiobjectif constituent un domaine de recherche et de développement très actif.

D'une part, les métaheuristiques représentent une classe de méthodologies de résolution approchées qui sont applicables à une large variété de problèmes d'optimisation. Elles sont indépendantes du problème à résoudre, et doivent donc être ajustées à la résolution d'un problème particulier, ceci par le biais d'un sous-ensemble de composants algorithmiques dépendants du problème traité. Les métaheuristiques sont des approches générales de haut niveau utilisées pour guider et diriger, de façon sous-jacente, la conception d'une méthode de résolution dédiée à un problème d'optimisation particulier. Elles comptent parmi les techniques les plus efficaces pour résoudre, en un temps de calcul raisonnable, des problèmes d'optimisation difficiles dans de nombreux domaines de la recherche opérationnelle.
Ces dernières années, il est devenu évident que le développement de métaheuristiques efficaces se base sur un processus d'ingénierie complexe qui combine les aspects de conception, d'implémentation et d'analyse expérimentale. La difficulté de ce processus est en partie due à la complexité des problèmes abordés et au grand nombre de degrés de liberté accordés aux chercheurs et aux praticiens confrontés à l'élaboration de tels algorithmes. Ce processus de développement requiert une méthodologie solide qui aborde les différentes questions survenant lors des phases de conception algorithmique, d'implémentation, ainsi que d'évaluation et d'analyse expérimentale. De plus, des études supplémentaires sont nécessaires pour comprendre quelles métaheuristiques se montrent les plus adaptées à tel ou tel type de problème, et pour comprendre les relations existantes entre les multiples composants algorithmiques, l'influence des paramètres, etc.

D'autre part, de nombreux problèmes d'optimisation issus du monde réel requièrent la considération simultanée de plusieurs critères, généralement en conflit les uns avec les autres. À chacun de

1. *Discrete multiobjective Optimization for Large scale Problems with Hybrid dIstributed techNiques.*

ces critères correspond une fonction objectif à optimiser (minimiser ou maximiser), qui relève de la modélisation du problème traité. De tels problèmes d'optimisation multiobjectif sont réputés pour être particulièrement difficiles à résoudre. En effet, contrairement au cas monoobjectif, il n'existe pas une solution optimale unique, mais un ensemble de solutions optimales, dites *Pareto optimales*. Une solution est Pareto optimale si l'amélioration à l'égard d'une des fonctions objectif entraîne invariablement une détérioration relativement à une autre fonction objectif. Ces solutions représentent les compromis entre les différentes fonctions objectif à optimiser. L'union des solutions Pareto optimales forme l'*ensemble Pareto optimal*, et l'image de cet ensemble dans l'espace objectif est appelé le *front Pareto*. Ainsi, l'optimisation multiobjectif s'intéresse aux particularités liées à l'existence de ces solutions optimales multiples, et aux méthodes de résolution dédiées à ce type de problèmes, bien souvent NP-difficiles.
La résolution d'un problème d'optimisation multiobjectif consiste à trouver la solution Pareto optimale qui répond au mieux aux préférences du décideur. L'optimisation multiobjectif est donc étroitement liée au domaine de la prise de décision multicritère. Les approches de résolution initiales consistaient à réduire le problème original en un problème monoobjectif. Néanmoins, depuis plusieurs années, une des questions fondamentales de l'optimisation multiobjectif s'apparente à l'identification de l'ensemble Pareto optimal, ou à une approximation de celui-ci pour des problèmes difficiles de grande taille. Dans ce contexte, les métaheuristiques en général, et les algorithmes évolutionnaires en particulier, connaissent un intérêt grandissant. Dans la littérature du domaine, la résolution d'un problème multiobjectif à l'aide d'une métaheuristique est en effet communément entendue comme la recherche d'une approximation de l'ensemble Pareto optimal satisfaisant les conditions unanimes de convergence vers le front Pareto et de diversité uniforme. Ces deux caractéristiques contrôlent l'obtention de solutions proches, et bien distribuées le long du front Pareto, de sorte qu'aucune information utile ne soit perdue. Par rapport au cas monoobjectif, les difficultés supplémentaires liées à la résolution de problèmes d'optimisation multiobjectif résultent du fait qu'il n'existe pas d'ordre total entre les solutions, que la structure du front Pareto est variable d'un problème à l'autre, et que le nombre de solutions Pareto optimales croît en fonction de la taille du problème, en particulier selon le nombre d'objectifs.

Dans ce mémoire de thèse, nous nous intéressons aux problématiques liées à la résolution de problèmes d'optimisation multiobjectif à l'aide de métaheuristiques, et en particulier aux phases successives de conception, d'implémentation et d'analyse expérimentale de ce type d'approches. Par ailleurs, nous étudions deux problèmes d'optimisation combinatoires issus du domaine de la logistique, et à l'application de métaheuristiques pour leur résolution. Puis, suite à une étude approfondie des métaheuristiques pour l'optimisation multiobjectif, nous nous intéressons à la coopération de plusieurs métaheuristiques. Nous montrons pour finir qu'en plus de leur complexité intrinsèque et de leurs multiples fonctions objectif, beaucoup de problèmes d'optimisation sont également sujets à toute sorte d'incertitude. Nous examinons, d'un point de vue métaheuristique et multiobjectif, comment prendre ces différentes sources d'incertitude en compte, et quelles en sont les répercussions sur le développement des méthodes de résolution. Les différents aspects couverts au cours de ce mémoire de thèse sont illustrés sur la figure 1.
Dans un premier temps, nous nous concentrons principalement sur le développement de métaheuristiques dédiées à l'approximation de l'ensemble Pareto optimal. Une vue unifiée est présentée pour la conception de métaheuristiques efficaces dédiées à la résolution de problèmes d'optimisation multiobjectif continus ou combinatoires. Un accent particulier est mis sur les composants de recherche spécifiques à l'aspect multiobjectif. Nous proposons ensuite une plateforme logicielle

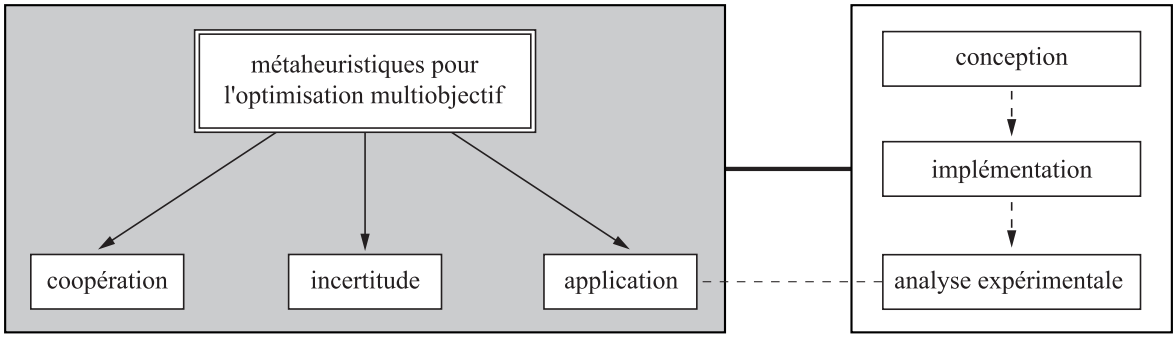


Fig. 1 – Interconnexion entre les principaux thèmes abordés au cours de ce manuscrit de thèse.

dédiée à l'implémentation flexible et réutilisable de telles méthodes : ParadisEO-MOEO. Il s'agit du module de la plateforme ParadisEO spécifiquement consacré aux métaheuristiques pour l'optimisation multiobjectif, et basé sur une séparation claire entre la méthode de résolution et le problème à traiter. Enfin, nous modélisons les deux problèmes traités ici en vue de leur résolution par métaheuristique, et nous menons une analyse expérimentale d'un ensemble de méthodes sur ces deux applications.

Dans un second temps, nous nous attardons sur le développement de métaheuristiques coopératives, séquentielles et parallèles, toujours dans le cadre de l'optimisation multiobjectif. En effet, les métaheuristiques sont considérées comme des méthodes de référence pour bon nombre de problèmes d'optimisation. Cependant, il est à présent devenu évident que se concentrer sur un seul type de métaheuristiques s'avère plutôt restrictif. Une combinaison appropriée de concepts provenant de différentes méthodologies permet dans bien des cas de produire un comportement plus efficace et une plus grande flexibilité lorsque des problèmes de grande envergure issus du monde réel sont traités. Les métaheuristiques hybrides sont des approches d'optimisation qui combinent différentes métaheuristiques, ou qui intègrent des techniques issues de l'intelligence artificielle ou de la recherche opérationnelle au sein d'une métaheuristique. Pour cela, nous proposons plusieurs schémas de coopération généraux, et validons leurs intérêts sur les deux problèmes d'optimisation multiobjectif combinatoire étudiés.

Dans un dernier temps, nous constatons qu'une grande partie des problèmes d'optimisation réels sont soumis à des incertitudes, tant au niveau des fonctions objectif qu'au niveau des paramètres environnementaux, ou encore des variables de décision. Ces différentes sources d'incertitude se doivent d'être prises en compte au cours de la modélisation et de la résolution du problème. D'un point de vue multiobjectif, ces incertitudes constituent un obstacle majeur, puisqu'elles empêchent l'identification de l'ensemble Pareto optimal. En dépit de leur flexibilité, les métaheuristiques sont très rarement explorées pour s'attaquer à des problèmes de nature stochastique. Pourtant, elles semblent être des candidates intéressantes. Nous pensons qu'avec un minimum d'ajustements, elles peuvent être appliquées à une large variété de problèmes d'optimisation sous incertitude, dont des problèmes multiobjectif. Nous examinons tout d'abord comment prendre en compte ces différentes sources d'incertitude, et nous nous intéressons pour cela aux approches fondées sur un échantillonnage. Ensuite, nous discutons de l'adaptation des métaheuristiques pour résoudre ce type de problèmes, et nous recensons les différentes alternatives envisageables.

Plan du manuscrit

Chapitre 1. Le premier chapitre introduit les bases de l'optimisation multiobjectif, connaissances requises pour la bonne compréhension des travaux présentés dans les chapitres ultérieurs. Ainsi, nous donnons les définitions essentielles, et discutons de différentes approches de résolution et de l'analyse de leurs performances. De plus, nous présentons les deux problèmes d'optimisation combinatoire multiobjectif qui vont nous intéresser par la suite. Il s'agit d'un problème d'ordonnancement de type Flowshop déjà étudié au sein de l'équipe DOLPHIN (Basseur, 2005; Lemesre, 2006), et d'un problème de tournées pour la première fois formulée de façon multiobjectif : le problème de Ring-Star, faisant référence au travail de Jozefowiez (2004).

Chapitre 2. Le second chapitre constitue le noyau de cette thèse. Il s'intéresse à la conception, à l'implémentation et à l'analyse expérimentale de métaheuristiques pour l'optimisation multiobjectif. Une unification de la conception de telles méthodes est proposée. Bien que cette vue unifiée soit commune à tout type de métaheuristique, l'accent est mis sur deux méthodologies particulières : les algorithmes évolutionnaires et les algorithmes de recherche locale. Diverses contributions algorithmiques sont également présentées. Nous dévoilons ensuite la traduction de ce modèle de conception en une plateforme logicielle dédiée à l'implémentation de métaheuristiques pour l'optimisation multiobjectif : ParadisEO-MOEO. Celle-ci fait suite à un travail de thèse réalisée dans l'équipe DOLPHIN (Cahon, 2005). Enfin, nous fournissons une étude comparative à propos des performances d'un ensemble de méthodes pour la résolution des deux problèmes traités ici : le problème de Flowshop et le problème de Ring-Star. Ces deux applications nous permettent par ailleurs d'analyser expérimentalement l'influence des composants de recherche sur le comportement général de ces algorithmes.

Chapitre 3. Au cours du troisième chapitre, nous nous intéressons aux métaheuristiques hybrides pour l'optimisation multiobjectif. Deux modèles de coopération sont proposés. Le premier se base sur la complémentarité de deux métaheuristiques, exécutées en séquence, pour accomplir des tâches contradictoires : l'amélioration de solutions existantes (intensification) et la découverte de nouvelles régions à explorer (diversification). Le deuxième schéma d'hybridation se base sur l'exécution parallèle de plusieurs métaheuristiques. Chacune d'elles s'emploie à trouver une approximation de l'ensemble Pareto optimal localisé dans une sous-région de l'espace objectif ; ce dernier étant divisé à l'aide de points de référence multiples.

Chapitre 4. Le dernier chapitre discute de la prise en compte de l'incertitude au sein de problèmes d'optimisation multiobjectif. Nous analysons différentes options pour la gestion des sources d'incertitude, et la manière dont ces options affectent la conception de métaheuristiques. Un bilan et une classification des approches existantes sont proposés. Nous présentons ensuite différentes contributions méthodologiques, en termes de résolution algorithmique et d'analyse de performances. Enfin, nous étudions la résolution du problème de Flowshop multiobjectif avec durées d'exécutions stochastiques, et soulevons de nouvelles perspectives à propos des métaheuristiques pour l'optimisation multiobjectif en environnement incertain.

Nous conclurons le manuscrit en synthétisant les principales contributions présentées, et en dénombrant de nouvelles voies à explorer suite à ce travail de thèse.

Chapitre 1

Optimisation multiobjectif

Ce premier chapitre a pour but d'introduire les prérequis nécessaires à la bonne compréhension de cette thèse. Ainsi, dans un premier temps, nous présenterons le contexte de l'optimisation multiobjectif, les principales définitions, et surtout les problématiques essentielles liées à ce domaine de recherche, tant au niveau des approches de résolution qu'à celui de l'évaluation des performances. Dans un second temps, nous présenterons les deux problèmes d'optimisation combinatoire multiobjectif auxquels nous allons particulièrement nous intéresser par la suite, à savoir le problème de Flowshop, et le problème de Ring-Star.

Sommaire

Introduction

L'optimisation multiobjectif est un domaine fondamental de l'aide à la décision multicritère, auquel de nombreux milieux scientifiques et industriels se doivent de faire face. Au cours des deux dernières décennies, un très grand nombre de travaux, à la fois théoriques et appliqués, ont été publiés dans ce domaine. La résolution d'un problème d'optimisation multiobjectif consiste à déterminer la solution correspondant au mieux aux préférences du décideur parmi les solutions de bon compromis. L'une des questions les plus difficiles est donc liée à l'identification de l'ensemble Pareto optimal, ou d'une approximation de celui-ci pour des problèmes complexes.
Ainsi, de nombreux domaines applicatifs se doivent de répondre à l'optimisation multiobjectif. En particulier, beaucoup de problèmes de logistique rencontrés dans l'industrie sont de nature multiobjectif. La logistique et les transports sont au centre de nombreuses préoccupations économiques et pratiques, notamment dans la région Nord-Pas-de-Calais. Il n'est donc pas étonnant que ce type de problème soit plus qu'abondamment étudié en recherche opérationnelle. Au sein de cette thèse, nous allons nous intéresser à deux problèmes d'optimisation multiobjectif issus de la logistique et des transports en général, et de l'ordonnancement et du routage en particulier. Il s'agit d'un problème d'ordonnancement de type Flowshop, présenté sous une forme à deux et à trois fonctions objectif, et du problème de Ring-Star, généralisation d'un problème de tournées en un problème biobjectif.

Le chapitre est organisé de la façon suivante. Tout d'abord, un ensemble de définitions pour l'optimisation multiobjectif est donné dans la section 1.1. Puis, dans la section 1.2, nous présentons une classification des différentes approches de résolution. Ensuite, la section 1.3 traite de l'évaluation des performances dans le cadre de l'optimisation multiobjectif. Enfin, dans la section 1.4, nous présentons deux problèmes d'optimisation combinatoire multiobjectif pratiques et complexes que nous traiterons par la suite.

1.1 Définitions

Cette section couvre les principaux concepts, définitions et notations liés à l'optimisation multiobjectif, tels que la relation de dominance Pareto, l'optimalité Pareto, l'ensemble Pareto optimal, et le front Pareto. De nombreuses définitions sont tirées du livre de Ehrgott (2005).

1.1.1 Formulation générale d'un problème d'optimisation multiobjectif

Définition 1.1 (Problème d'optimisation multiobjectif) Un problème d'optimisation multiobjectif (*multiobjective optimization problem, MOP*) peut être défini comme suit :

$$(MOP) = \left\{ \begin{array}{ll} \text{“min”} & f(x) = (f_1(x), f_2(x), \ldots, f_n(x)) \\ \text{tel que :} & x \in X \end{array} \right. \tag{1.1}$$

où n est le nombre d'objectifs ($n \geq 2$), X est l'ensemble des solutions réalisables dans l'*espace décisionnel*, et $x = (x_1, x_2, \ldots, x_k) \in X$ est un vecteur représentant les variables de décision. Dans le cas combinatoire, X est un ensemble discret. À chaque solution $x \in X$ est associé un vecteur objectif $z \in Z$ sur la base d'un vecteur de fonctions $f : X \to Z$ tel que $z = (z_1, z_2, \ldots, z_n) = f(x) = (f_1(x), f_2(x), \ldots, f_n(x))$, où $Z = f(X)$ représente l'ensemble des points réalisables de

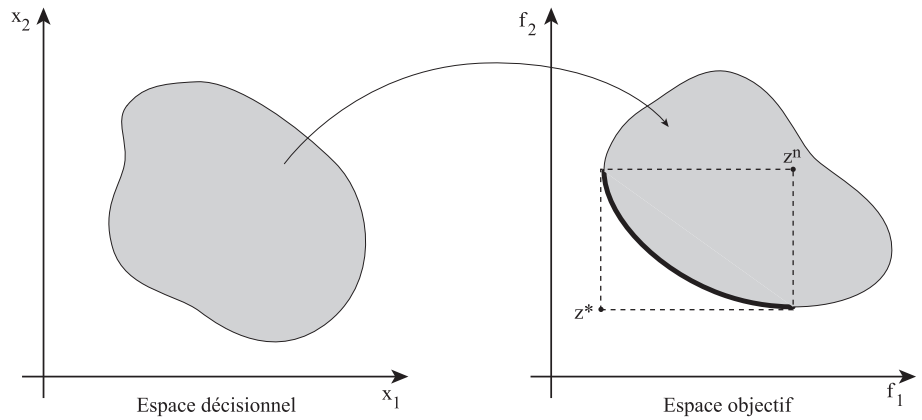


Fig. 1.1 – Espace décisionnel et espace objectif d'un problème d'optimisation multiobjectif (exemple avec deux variables de décisions et deux fonctions objectif).

l'*espace objectif* (Fig. 1.1). Sans perte de généralité, nous supposerons par la suite que $Z \subseteq \mathbb{R}^n$ et que toutes les fonctions objectif sont à minimiser.

1.1.2 Notions de dominance Pareto et d'optimalité

Dans le domaine de l'optimisation multiobjectif, le décideur évalue généralement une solution par rapport à chacun des critères, et se positionne donc naturellement dans l'espace objectif. Néanmoins, contrairement au cas monoobjectif où il existe un ordre total parmi les solutions réalisables, il n'existe généralement pas de solution qui serait à la fois optimale pour chaque objectif, étant donnée la nature conflictuelle de ces derniers. Ainsi, une *relation de dominance*, d'ordre partiel, est généralement définie.

Définition 1.2 (Dominance Pareto) Un vecteur objectif $z \in Z$ domine un vecteur objectif $z' \in Z$ ssi $\forall i \in \{1, 2, \ldots, n\}$, $z_i \leq z'_i$ et $\exists j \in \{1, 2, \ldots, n\}$ tel que $z_j < z'_j$. Cette relation sera notée $z \succ z'$. Par extension, une solution $x \in X$ domine une solution $x' \in X$, noté $x \succ x'$, ssi $f(x) \succ f(x')$.

Définition 1.3 (Vecteur non-dominé) Un vecteur objectif $z \in Z$ est non-dominé ssi $\forall z' \in Z$, $z' \not\succ z$ (Fig. 1.2).

Définition 1.4 (Solution Pareto optimale) Une solution $x \in X$ est Pareto optimale (ou non-dominée) ssi $\forall x' \in X$, $x' \not\succ x$.

Ainsi, toute solution Pareto optimale peut être considérée comme optimale puisqu'aucune amélioration n'est possible sur la valeur d'un objectif sans en dégrader celle d'un autre. La notion de Pareto optimalité, initialement utilisée en économie et dans les sciences de management, prend ses racines dans les travaux de Edgeworth (1881) et Pareto (1896). Ces solutions Pareto optimales forment l'*ensemble Pareto optimal*, noté X_E. L'image de cet ensemble dans l'espace objectif est le *front Pareto*, et sera noté Z_N.

Définition 1.5 (Ensemble Pareto optimal) Étant donné un MOP(f,X), l'*ensemble Pareto optimal* est défini comme suit : $X_E = \{x \in X | \nexists x' \in X, x' \succ x\}$.

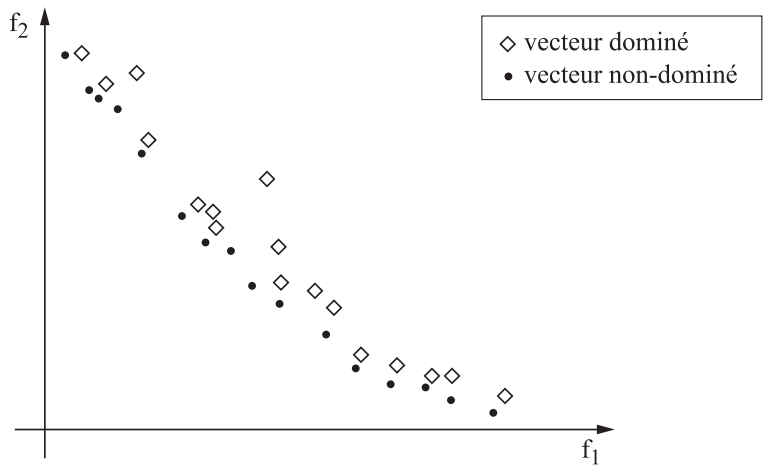


Fig. 1.2 – Vecteurs dominés et non-dominés dans l'espace objectif.

Définition 1.6 (Front Pareto) Étant donné un MOP(f,X) et son ensemble Pareto optimal X_E, le *front Pareto* est défini comme suit : $Z_N = \{f(x)|x \in X_E\}$.

1.1.3 Autres notions de dominance

Il existe des relations de dominance différentes de la dominance Pareto, comme la dominance faible, la dominance stricte et l'ϵ-dominance (Helbig et Pateva, 1994; Laumanns *et al.*, 2002). Elles sont définies ci-dessous.

Définition 1.7 (Dominance faible) Un vecteur objectif $z \in Z$ domine *faiblement* un vecteur objectif $z' \in Z$ ssi $\forall i \in \{1, 2, \ldots, n\}$, $z_i \leq z'_i$. Cette relation sera notée $z \succeq z'$.

Définition 1.8 (Dominance stricte) Un vecteur objectif $z \in Z$ domine *strictement* un vecteur objectif $z' \in Z$ ssi $\forall i \in \{1, 2, \ldots, n\}$, $z_i < z'_i$. Cette relation sera notée $z > z'$.

Définition 1.9 (ϵ-dominance) Un vecteur objectif $z \in Z$ *ϵ-domine* un vecteur objectif $z' \in Z$, avec $\epsilon > 1$, ssi $\forall i \in \{1, 2, \ldots, n\}$, $z_i \leq \epsilon \cdot z'_i$. Cette relation sera notée $z \succeq_\epsilon z'$.

Définition 1.10 (ϵ-dominance additive) Un vecteur objectif $z \in Z$ *ϵ-domine* de façon additive un vecteur objectif $z' \in Z$, avec $\epsilon > 0$, ssi $\forall i \in \{1, 2, \ldots, n\}$, $z_i \leq \epsilon + z'_i$ (Fig. 1.3). Cette relation sera notée $z \succeq_{\epsilon+} z'$.

Par extension, on dira qu'une solution $x \in X$ domine faiblement, domine strictement ou ϵ-domine une solution $x' \in X$ si la relation de dominance est vérifiée pour les vecteurs objectif correspondants (respectivement $f(x)$ et $f(x')$).

1.1.4 Bornes du front Pareto

Maintenant, supposons que l'optimum soit connu pour chaque fonction objectif, alors nous pouvons définir la notion de point idéal, de point utopique et de point nadir.

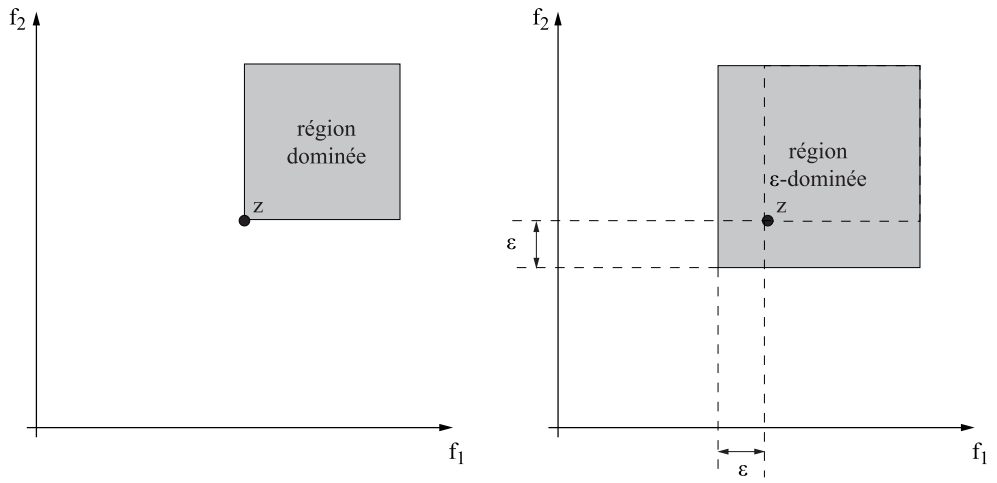


Fig. 1.3 – Illustration de la notion d'ϵ-dominance additive.

Définition 1.11 (Point idéal) Le *point idéal* $z^* = (z_1^*, z_2^*, \ldots, z_n^*)$ est le vecteur qui minimise chaque fonction objectif individuellement, $z_i^* = \min_{x \in X} f_i(x)$ pour $i \in \{1, 2, \ldots, n\}$.

Bien sûr, ce vecteur idéal est rarement réalisable en pratique car les objectifs sont souvent en conflit les uns avec les autres.

Définition 1.12 (Point utopique) Le *point utopique* $z^{**} = (z_1^{**}, z_2^{**}, \ldots, z_n^{**})$ est le vecteur objectif irréalisable dont les composantes sont formées par $z_i^{**} = z_i^* - \epsilon_i$ pour $i \in \{1, 2, \ldots, n\}$, où z^* est le point idéal et ϵ est un vecteur de valeurs scalaires strictement positives relativement petites mais significatives.

Définition 1.13 (Point nadir) Le *point nadir* $z^n = (z_1^n, z_2^n, \ldots, z_n^n)$ est le vecteur qui correspond au maximum de chaque fonction objectif parmi les solutions de l'ensemble Pareto optimal, $z_i^n = \max_{x \in X_E} f_i(x)$ pour $i \in \{1, 2, \ldots, n\}$.

Ce point nadir est bien plus difficile à calculer que le point idéal, notamment lorsque le nombre de fonctions objectif est strictement supérieur à deux. Le point idéal z^* et le point nadir z^n sont représentés graphiquement sur la figure 1.1.

1.2 Approches de résolution

Cette section présente les différentes approches de résolution de l'optimisation multiobjectif. Tout d'abord, une classification sur la façon de résoudre le problème traité du point de vue du décideur est présentée. Ensuite, nous délimitons les différentes formes que peut prendre l'ensemble Pareto optimal. Puis nous finissons par une taxinomie des principales méthodologies de résolution d'un point de vue algorithmique. Une classification de toutes ces questions essentielles pour la résolution de problèmes d'optimisation multiobjectif est donnée dans la figure 1.4.

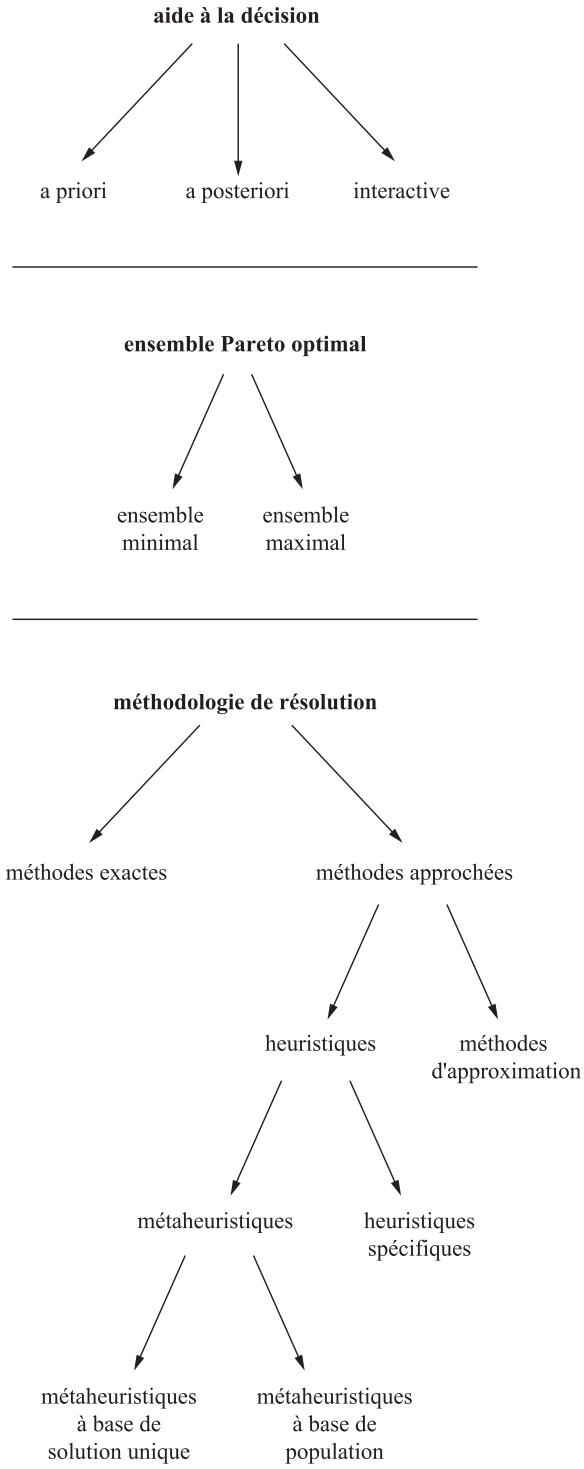


Fig. 1.4 – Classification des approches de résolution pour l'optimisation multiobjectif.

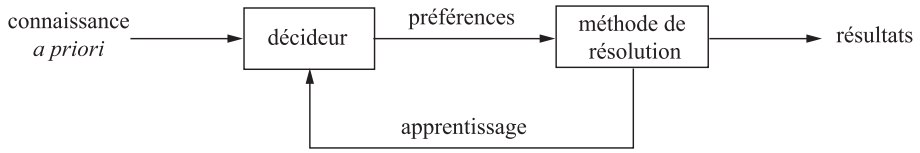


Fig. 1.5 – Approche de résolution interactive : coopération progressive entre le décideur et la méthode de résolution.

1.2.1 Optimisation multiobjectif et aide à la décision

Résoudre un problème d'optimisation multiobjectif consiste à trouver la solution Pareto optimale qui correspond le mieux aux préférences du décideur. L'une des questions fondamentales lors de la résolution d'un tel problème est donc étroitement liée à la coopération entre le décideur et la méthode de résolution. Plusieurs scénarios existent à propos du rôle que peut jouer le décideur dans un processus de prise de décision.

- ***A priori***. Dans ce cas, le décideur fournit des connaissances ou des préférences sur le problème à résoudre avant le processus d'optimisation, ceci en vue d'aider la méthode de résolution dans sa recherche. En pratique, ceci se traduit fréquemment en la transformation du problème d'optimisation multiobjectif original en un problème monoobjectif, et peut donc être résolu par une méthode classique dont une seule exécution fournira la solution recherchée. Néanmoins, la modélisation des préférences du décideur n'est pas une tâche facile et doit être prise en compte durant l'étape de résolution. Par ailleurs, il se peut que le décideur ne soit pas satisfait de la solution trouvée et qu'il doive donc relancer le processus par le biais d'une autre formulation de ses préférences.
- ***A posteriori***. Ici la méthode de résolution vise à trouver la totalité des solutions Pareto optimales, ou une bonne approximation de cet ensemble. Ceci permet au décideur d'avoir une connaissance complète du front Pareto. Il peut alors choisir, parmi cet ensemble de solutions, celle qui lui semble la plus appropriée au vue de ses préférences. Ainsi, il n'est plus nécessaire de modéliser les préférences du décideur, mais il faut désormais fournir un ensemble de solutions, ce qui peut s'avérer coûteux en pratique. Par ailleurs, le nombre de solutions obtenues peut rendre l'ensemble Pareto optimale difficile à analyser pour le décideur.
- **Interactive**. Lors d'une approche interactive, il existe une coopération directe et progressive entre le décideur et la méthode de résolution (Fig. 1.5). Ainsi, à partir de connaissances acquises lors de la résolution, le décideur peut définir ses préférences de façon compréhensible. Ce processus est itéré plusieurs fois jusqu'à la satisfaction du décideur. Toutefois, cette approche nécessite la présence du décideur tout au long du processus de recherche.

Chaque approche possède ses forces et ses faiblesses. Le choix d'un type de méthode dépend des propriétés du problème à résoudre et des habilités du décideur. Au cours de ce manuscrit, nous nous placerons dans le cadre des méthodes non-interactives, et principalement des méthodes *a posteriori*. Dans ce dernier type d'approche, deux phases sont considérées : la phase d'optimisation et la phase d'aide à la décision. Seule la première phase sera traitée ici, et sera abusivement considérée comme la résolution du problème traité. Toutefois, notez que les méthodes non-interactives peuvent également s'avérer utiles lors de la conception d'une approche interactive. Par exemple, lorsque cela est possible, la visualisation du front Pareto offre une connaissance précieuse au décideur en vue de l'aider à formuler ses préférences.

1.2.2 Classification des ensembles Pareto optimaux

Nous considérons donc que le but de l'optimisation est de fournir l'ensemble Pareto optimal (ou une bonne approximation de cet ensemble) au décideur afin que celui-ci procède à la sélection de la, ou des solutions qui correspondent à ses préférences. Aussi, étant donné que deux solutions peuvent posséder les mêmes valeurs pour chacun des objectifs, une question est de savoir s'il est intéressant de conserver deux solutions Pareto optimales équivalentes.

Définition 1.14 (Solutions équivalentes) Deux solutions $x, x' \in X$ sont équivalentes ssi $f(x) = f(x')$.

Ainsi, la classification suivante peut être introduite pour l'ensemble Pareto optimal (Hansen, 1979).

Définition 1.15 (Ensemble complet) Étant donné un MOP(f,X), son ensemble Pareto optimal X_E et son front Pareto Z_N, un *ensemble complet* est un ensemble $\widehat{X} \in X_E$ tel que, pour chaque vecteur non-dominé $z \in Z_N$, il existe au moins une solution $x \in \widehat{X}$ pour laquelle $f(x) = z$.

Définition 1.16 (Ensemble complet minimal) Un *ensemble complet minimal* est un ensemble complet sans solutions équivalentes.

Définition 1.17 (Ensemble complet maximal) Un *ensemble complet maximal* est un ensemble complet contenant toutes les solutions équivalentes.

À l'image de la finalité rencontrée dans le cadre de l'optimisation monoobjectif, où une seule solution optimale est généralement fournie, obtenir un ensemble complet minimal est le but principal de l'optimisation multiobjectif, à l'exception de quelques applications particulières. Par ailleurs, notez que dans le cas où le but est uniquement de fournir une image de l'ensemble Pareto optimal dans l'espace objectif au décideur, il s'avère inutile de conserver les solutions équivalentes.

1.2.3 Méthodologies de résolution

En fonction de la complexité du problème d'optimisation à résoudre, plusieurs types de méthode peuvent être envisagés. Dans un premier temps, celles-ci sont brièvement présentées. Une attention particulière est portée aux métaheuristiques, auxquelles on va essentiellement s'intéresser tout au long de ce manuscrit. Enfin, quelques remarques d'ordre général sont données à propos du but principal des méthodes de résolution approchées dans le cadre de l'optimisation multiobjectif.

1.2.3.1 Classification

De façon générale, deux types de méthode de résolution de problèmes d'optimisation peuvent être distingués : les *méthodes exactes* et les *méthodes approchées* (Fig. 1.4). Les méthodes exactes obtiennent des solutions dont l'optimalité est garantie, mais sont généralement très coûteuses en temps d'exécution pour des problèmes difficiles. Au contraire, les méthodes approchées visent à générer des solutions de haute qualité en un temps de calcul raisonnable, mais il n'existe aucune garantie de trouver la solution optimale. Ces méthodes approchées sont elles-mêmes divisées en

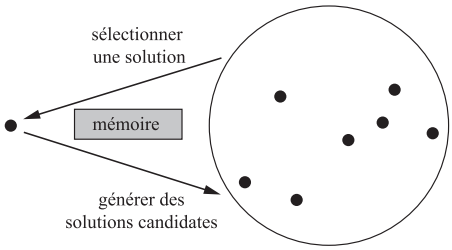


Fig. 1.6 – Illustration des principes généraux d'une métaheuristique à base de solution unique (S-META).

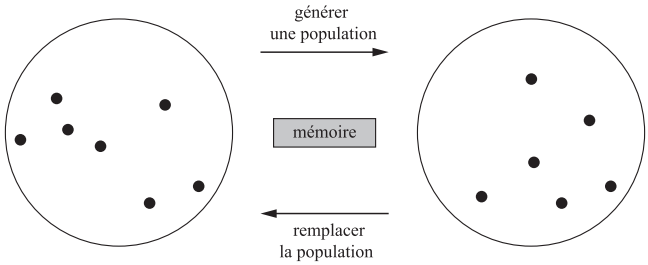


Fig. 1.7 – Illustration des principes généraux d'une métaheuristique à base de population (P-META).

deux sous-catégories : les *algorithmes d'approximation* et les *heuristiques*. Contrairement aux heuristiques, les algorithmes d'approximation garantissent la qualité de la solution trouvée par rapport à l'optimal. Enfin, il existe encore deux sous-classes de méthodes heuristiques : les heuristiques spécifiques à un problème donné, et les *métaheuristiques*, qui seront considérées dans le cadre de cette thèse.

1.2.3.2 Métaheuristiques

Les métaheuristiques sont des algorithmes pouvant être appliqués à la résolution de presque tout type de problème d'optimisation. Elles peuvent être vues comme des méthodologies de haut niveau servant à guider la conception d'heuristiques implicitement dédiées à la résolution d'un problème spécifique. Elles sont donc composées d'éléments génériques ou invariants, ainsi que d'éléments spécifiques au problème considéré, tels que la représentation ou l'évaluation d'une solution.

À l'inverse des méthodes exactes, les métaheuristiques permettent de s'attaquer à des problèmes complexes de grande taille en délivrant des solutions satisfaisantes en un temps de calcul raisonnable. Néanmoins, il n'existe pas de garantie quant à leur optimalité. Durant les vingt dernières années, les métaheuristiques ont reçu un intérêt grandissant et ont montré leur efficacité dans de vastes domaines d'application en résolvant de nombreux problèmes d'optimisation (Talbi, 2009). Deux types de métaheuristique peuvent être distingués : les *métaheuristiques à base de solution unique* (S-META) et les *métaheuristiques à base de population* (P-META). Les S-META (tels que les algorithmes de recherche locale, de recherche tabou, de recuit simulé, etc.) manipulent et transforment une seule solution durant le processus de recherche, alors que dans les P-META (algorithmes évolutionnaires, algorithmes à essaim de particules, etc.), un ensemble de solutions, appelé *population*, évolue en parallèle. Les principes généraux d'une S-META et d'une P-META sont respectivement illustrés sur les figures 1.6 et 1.7 (Talbi, 2009).

En termes de conception, deux critères contradictoires sont à prendre en compte lors du développement d'une métaheuristique : l'exploration de l'espace de recherche (*diversification*), et l'exploitation des meilleures solutions trouvées (*intensification*). Les S-META sont plutôt axées sur l'exploitation de l'espace de recherche. Les régions prometteuses sont explorées localement dans l'espoir de trouver de meilleurs solutions. Les P-META sont généralement plutôt exploratoires et permettent une meilleure diversification de l'espace de recherche.

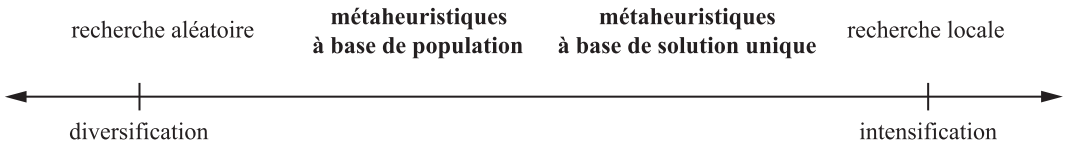


Fig. 1.8 – Deux critères contradictoires lors de la conception d'une métaheuristique : exploration (diversification) et exploitation (intensification). En général les métaheuristiques à base de solution unique sont plutôt orientées vers l'exploitation, alors que les métaheuristiques à base de population sont plutôt exploratoires (Talbi, 2009).

1.2.3.3 Méthodes pour l'optimisation multiobjectif

Dans le cadre de l'optimisation multiobjectif, les problèmes rencontrés sont souvent NP-difficiles, ce qui rend l'utilisation de méthodes exactes impraticable pour des instances de grande taille. En effet, si un problème monoobjectif donné est connu comme NP-difficile, alors sa contrepartie multiobjectif le sera aussi. Par ailleurs, pour de nombreux problèmes monoobjectif solubles en un temps polynomial, le problème multiobjectif correspondant est toutefois NP-difficile (Ehrgott, 2005). Ceci a pour effet de restreindre l'application de méthodes exactes et d'encourager l'utilisation d'heuristiques et de métaheuristiques. De plus, les méthodes exactes sont généralement limitées à des problèmes de petite taille et à deux fonctions objectif. On peut par exemple citer la méthode deux-phases, proposée par Ulungu et Teghem (1995), et plus tard améliorée par Lemesre *et al.* (2007a), ainsi que la méthode PPM (Lemesre *et al.*, 2007b), et k-PPM (Dhaenens *et al.*, 2010) pour des problèmes à plus de deux objectifs.
Afin de trouver une approximation de l'ensemble Pareto optimal, les P-META, et en particulier les algorithmes évolutionnaires, sont très couramment utilisées car elles sont naturellement conçues pour trouver un ensemble de solutions en une seule exécution (Deb, 2001; Coello Coello *et al.*, 2007). L'optimisation évolutionnaire multiobjectif (*evolutionary multiobjective optimization, EMO*) est un champ de recherche et de développement très actif, donnant lieu à un nombre grandissant de publications [1] et à une conférence bi-annuelle spécifique depuis 2001 (Zitzler *et al.*, 2001a; Fonseca *et al.*, 2003; Coello Coello *et al.*, 2005; Obayashi *et al.*, 2007; Ehrgott *et al.*, 2009).

L'application de métaheuristiques pour la résolution d'un problème d'optimisation multiobjectif implique de trouver une approximation de l'ensemble Pareto optimal. Pour une métaheuristique A dédiée à la résolution d'un problème donné, l'approximation courante forme un ensemble potentiellement Pareto optimal, et est constituée de solutions potentiellement Pareto optimales.

Définition 1.18 (Solution potentiellement Pareto optimale) Soit X_A l'ensemble des solutions de X visitées par un algorithme A : $X_A \subset X$. Une solution $x \in X_A$ est *potentiellement Pareto optimale* ssi $\forall x' \in X_A$, $x' \not\succ x$.

De façon abusive, on parlera également de solution non-dominée pour désigner une solution potentiellement Pareto optimale. L'*ensemble potentiellement Pareto optimal*, et donc l'approximation courante de l'algorithme, est l'union de ces solutions non-dominées.
Cette approximation doit contenir un ensemble de solutions dont l'image dans l'espace objectif est à la fois proche du front Pareto et uniformément diversifiée le long de la frontière. Ceci a

1. Voir la liste de références maintenue sur le sujet par C. A. Coello Coello à l'URL : `http://delta.cs.cinvestav.mx/~ccoello/EMOO/`.

pour but d'obtenir un échantillon représentatif et de ne pas se concentrer sur une sous-partie de l'espace objectif. Ainsi, lorsque l'on parle de diversité en optimisation multiobjectif, on se réfère généralement à l'espace objectif. Les deux buts de l'optimisation multiobjectif sont illustrés sur la figure 1.9. Maintenant, examinons les trois exemples donnés sur les figures 1.10, 1.11 et 1.12. Le premier exemple (Fig. 1.10) montre une approximation ayant les deux propriétés désirées de bonne convergence et de bonne diversité. L'approximation du deuxième exemple (Fig. 1.11) présente une très bonne répartition de solutions, mais les points sont loin du front Pareto. Enfin, le dernier exemple (Fig. 1.12) contient un ensemble de points très proches du front Pareto, mais certaines régions ne sont pas couvertes, ce qui provoque un perte d'information importante aux yeux du décideur.

La conception, l'implémentation et l'analyse de métaheuristiques pour l'optimisation multiobjectif est le sujet du chapitre 2.

1.3 Analyse de performances

Lors de l'évaluation et de la comparaison expérimentale des performances de métaheuristiques de façon rigoureuse, les points suivants doivent être abordés (Talbi, 2009). Tout d'abord, il est nécessaire de définir le but des expérimentations. Ensuite, des mesures, métriques ou indicateurs de performance doivent être sélectionnés pour obtenir les résultats, et une validation statistique de ces derniers doit être fournie. Enfin, ces résultats doivent être présentés de façon claire et être suivis d'une analyse tenant compte des objectifs fixés au départ des expérimentations. Le premier et le dernier points seront traités au gré des différentes applications menées au sein de ce travail. Cette section se concentre donc sur les mesures de performance ainsi que sur la validation statistique dans le cadre de l'optimisation multiobjectif.

1.3.1 Indicateurs de qualité

En optimisation multiobjectif, l'existence d'un ensemble de solutions et l'absence d'ordre total entre ces solutions rendent la mesure de qualité d'une approximation de l'ensemble Pareto optimal difficile. En outre, comme précédemment remarqué, à ceci vient s'ajouter la nécessité de mesurer la qualité d'une approximation à la fois en terme de convergence et de diversité. Ainsi, de nombreuses métriques ont été proposées pour mesurer la qualité d'un ensemble non-dominé, ou pour comparer deux approximations (Zitzler *et al.*, 2003, 2008).
Lors de nos expérimentations, nous allons principalement utiliser deux indicateurs de qualité afin de mesurer la performance des algorithmes testés : un indicateur basé sur la notion d'*hypervolume* et un indicateur basé sur la notion d'ϵ-*dominance*. De plus, une mesure de *contribution* sera également considérée pour un des algorithmes proposées dans la section 3.2. Après une classification générale des indicateurs de qualité, les trois indicateurs que nous allons considérer sont présentés ci-dessous.

1.3.1.1 Classification

Les indicateurs de qualité peuvent être classés selon différentes propriétés.

– **Arité**. Tout d'abord, il existe des indicateurs unaires et des indicateurs binaires. Un *indicateur unaire de qualité* est une fonction $I : \Omega \rightarrow \mathbb{R}$ qui affecte une valeur réelle à chaque approxi-

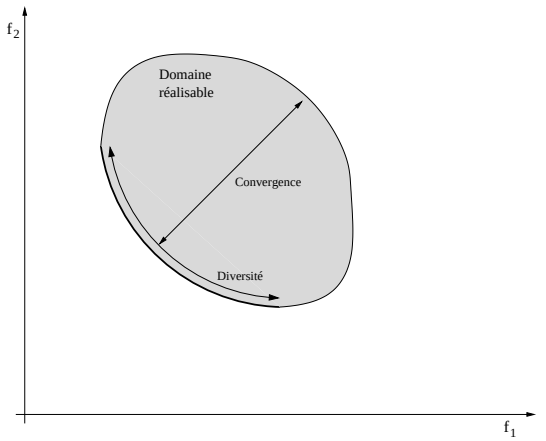


Fig. 1.9 – Les deux buts de l'optimisation multiobjectif : convergence et diversité.

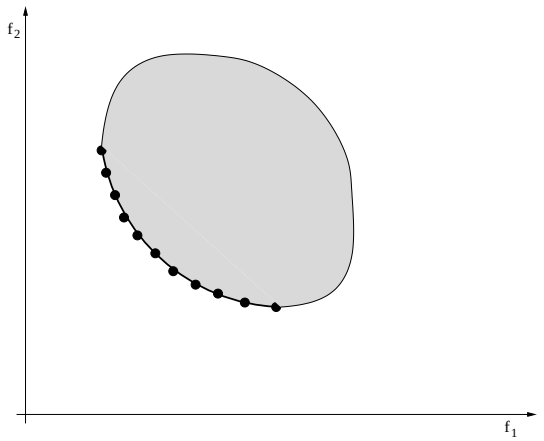


Fig. 1.10 – Exemple d'approximation du front Pareto de bonne qualité en terme de convergence et de diversité.

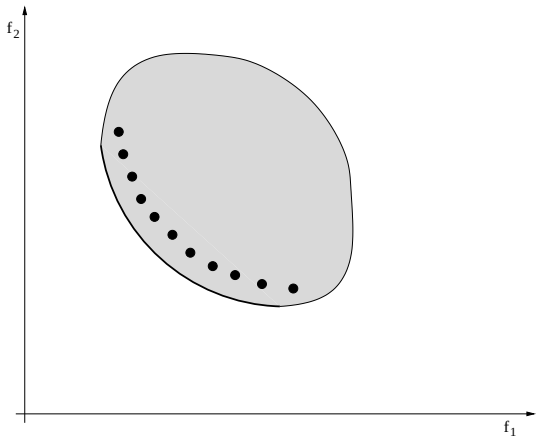


Fig. 1.11 – Exemple d'approximation du front Pareto de bonne qualité en terme de diversité, mais de mauvaise qualité en terme de convergence.

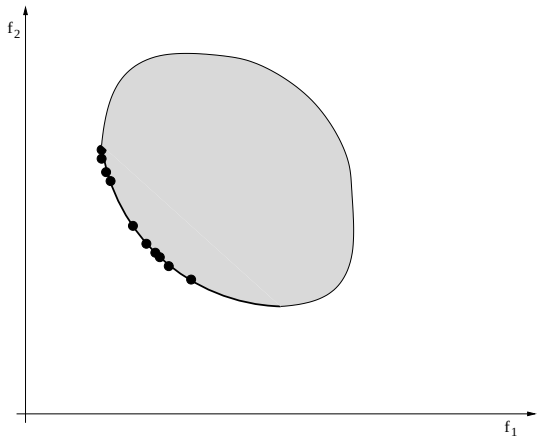


Fig. 1.12 – Exemple d'approximation du front Pareto de bonne qualité en terme de convergence, mais de mauvaise qualité en terme de diversité.

indicateur	arité	but de la mesure	paramètres	référence
contribution	binaire	convergence	–	Meunier *et al.* (2000)
coverage	binaire	convergence	–	Zitzler et Thiele (1999)
entropie	unaire	diversité	nombre de niches	Basseur *et al.* (2002)
epsilon	binaire	convergence, diversité	–	Zitzler *et al.* (2003)
proportion d'erreur	unaire	convergence, diversité	front Pareto exact	Veldhuizen et Lamont (2000)
distance générationelle	unaire	convergence	front Pareto exact	Veldhuizen et Lamont (2000)
hypervolume	unaire	convergence, diversité	point de référence	Zitzler et Thiele (1999)
hypervolume-différence	binaire	convergence, diversité	point de référence	Zitzler et Thiele (1999)
spacing	unaire	diversité	–	Schott (1995)
spread	unaire	diversité	–	Zitzler et Thiele (1999)
R1, R2, R3	binaire	convergence, diversité	ens. de référence, point idéal	Hansen et Jaszkiewicz (1998)

TABLE 1.1 – Caractéristiques d'un certain nombre d'indicateurs de qualité pour la mesure de performance en optimisation multiobjectif.

mation, où Ω représente l'ensemble de toutes les approximations possibles sur X. Ainsi, un indicateur de qualité I introduit un ordre total sur Ω. Cet ordre est sensé représenter la qualité des approximations. Supposons que l'on veuille comparer deux approximations arbitraires $A, B \in \Omega$, provenant par exemple de l'exécution de deux méthodes de résolution différentes sur un même problème. La différence entre leurs I-valeurs $I(A)$ et $I(B)$ mesure la différence de qualité entre ces deux ensembles. Ceci n'est pas uniquement valable dans le cas où les solutions de A sont toutes dominées par au moins un élément de B, mais également lorsque ces deux ensembles semblent *a priori* incomparables. Par extension, un *indicateur binaire de qualité* est une fonction $I : \Omega \times \Omega \rightarrow \mathbb{R}$ mesurant la qualité d'une approximation par rapport à une autre. Notez qu'un grand nombre d'indicateurs binaires peut être généralisé en indicateurs unaires de la façon suivante : $I(A) = I(A, R)$, où $R \in \Omega$ est un ensemble de référence. C'est le cas par exemple de l'indicateur hypervolume-différence, et de la famille des indicateurs epsilon, comme nous le verrons par la suite.

– **But de la mesure**. Les indicateurs de qualité peuvent également être distingués selon le but de leur mesure : la convergence vers le front Pareto, ou la diversité des solutions le long du front Pareto. Les indicateurs mesurant la convergence évaluent la qualité de la, ou des approximations en termes de proximité par rapport au front Pareto. Les indicateurs de diversité mesurent l'uniformité de la distribution des solutions, dans l'espace objectif, en termes de dispersion et d'extension. Enfin, les indicateurs hybrides portent à la fois sur la convergence et la diversité.
– **Paramètres**. De nombreux indicateurs de qualité nécessitent également la définition de certains paramètres. Parmi ces indicateurs, on peut distinguer ceux qui requièrent certaines connaissances exactes (front Pareto exact, point idéal, etc.) et ceux qui requièrent des informations de référence fournies par l'utilisateur (ensemble de référence, point de référence, etc.). Certains paramètres exacts, tels que le front Pareto, ne sont pas toujours disponibles et sont parfois impossibles à calculer. Néanmoins, lorsqu'une telle information est disponible, il s'avère intéressant de l'utiliser à bon escient, à l'aide d'indicateurs de qualité adaptés.

Le tableau 1.1 recense un ensemble non exhaustif d'indicateurs de qualité et certaines de leurs propriétés : arité, but de la mesure et paramètres.

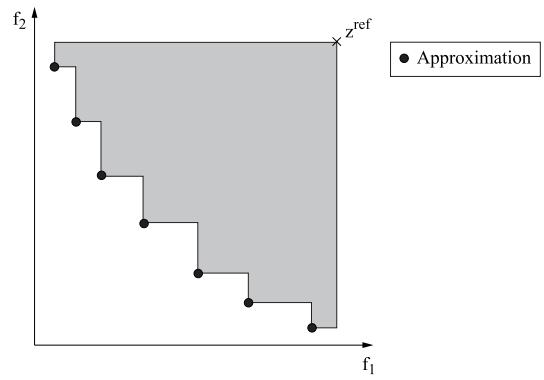


Fig. 1.13 – Indicateur hypervolume (I_H).

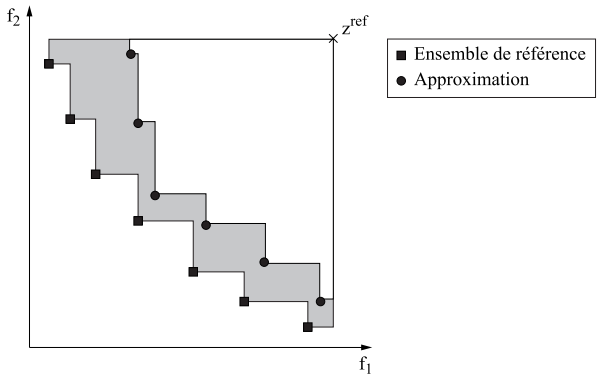


Fig. 1.14 – Indicateur hypervolume-différence (I_H^-).

1.3.1.2 Indicateur hypervolume

L'indicateur hypervolume (I_H), proposé par Zitzler et Thiele (1999), mesure le volume de la portion multidimensionnelle de l'espace objectif faiblement dominée par une approximation $A \in \Omega$ (Fig. 1.13). Cet indicateur unaire nécessite la spécification d'un point de référence z^{ref}. Ce dernier doit au moins être faiblement dominé par toutes les solutions de l'approximation considérée.
Une variante de cet indicateur mesure la différence, en terme d'hypervolume, entre une approximation $A \in \Omega$ et un ensemble de référence R (Fig. 1.14). Ceci peut être vu comme le volume faiblement dominé par R et non par A. C'est sous cette deuxième forme que nous allons utiliser l'indicateur hypervolume lors de nos expérimentations. Notez que contrairement à l'indicateur original, les plus petites valeurs correspondent ici à une qualité supérieure. Cet indicateur *hypervolume-différence* sera noté (I_H^-).

$$I_H^-(A) = I_H(R) - I_H(A). \tag{1.2}$$

Sous ses deux formes (I_H et I_H^-), l'indicateur hypervolume est un des rares indicateurs mesurant à la fois la qualité d'une approximation en terme de convergence et de diversité. Néanmoins, le temps de calcul est élevé : le meilleur algorithme connu est de complexité exponentiellement proportionnelle au nombre de fonctions objectif. Par ailleurs, il est à noter que cet indicateur est sensible à l'échelle des fonctions objectif et au choix du point de référence. Par conséquent, les magnitudes de toutes les fonctions objectif se doivent d'être normalisées, ceci afin de ne pas privilégier une fonction objectif au détriment d'une autre.

1.3.1.3 Indicateur epsilon

La famille des indicateurs epsilon a été introduite par Zitzler *et al.* (2003), et se base sur la notion d'efficacité epsilon introduite par Helbig et Pateva (1994). Elle comprend une version multiplicative et une version additive, toutes deux sous une forme unaire et sous une forme binaire. La version additive est celle que nous allons utiliser. Tout d'abord, l'*indicateur epsilon additif binaire* ($I_{\epsilon+}$) donne le facteur additif minimal par lequel une approximation $A \in \Omega$ doit être translatée dans l'espace objectif pour dominer faiblement une approximation $B \in \Omega$.

$$I_{\epsilon+}(A, B) = \min_{\epsilon \in \mathbb{R}} \{\forall x \in B, \exists x' \in A : x' \preceq_{\epsilon+} x\}. \tag{1.3}$$

Par extension, l'*indicateur epsilon additif unaire* ($I^1_{\epsilon+}$) peut être défini comme suit.

$$I^1_{\epsilon+}(A) = I_{\epsilon+}(A, R), \tag{1.4}$$

où R est un ensemble de référence. Cet indicateur est à minimiser. Par ailleurs, une $I^1_{\epsilon+}$-valeur inférieure ou égale à 0 implique que l'approximation considérée domine faiblement l'ensemble de référence R. L'indicateur epsilon semble à première vue dédié à la mesure de la qualité d'une approximation en terme de convergence. Cependant, entre deux approximations de qualité similaire en terme de convergence, l'ensemble le plus diversifié est privilégié. Enfin, tout comme l'indicateur hypervolume, $I^1_{\epsilon+}$ est sensible à l'échelle des fonctions objectif.

1.3.1.4 Indicateur contribution

La mesure de *contribution* (Meunier *et al.*, 2000) est un indicateur binaire permettant la comparaison de deux ensembles $A, B \in \Omega$. Soit $S^\star$ l'ensemble des solutions non-dominées de $A \cup B$. La contribution de A sur B, donnée par $I_C(A, B)$, évalue la proportion de solutions représentées par A dans $S^\star$. Soit W_A l'ensemble des solutions de A qui dominent au moins une solution de B. Soit N_A l'ensemble des solutions non comparables de A (c'est-à-dire les solutions qui sont ni dominantes, ni dominées par rapport à B). Alors la mesure de contribution peut être définie comme suit.

$$I_c(A, B) = \frac{\frac{|A \cap B|}{2} + |W_A| + |N_A|}{|S^\star|} \tag{1.5}$$

Ainsi, si $f(A) = f(B)$, alors $I_C(A, B) = I_C(B, A) = 0.5$. Si chaque solution de A est dominée par au moins une solution de B, alors $I_C(A, B) = 0$. Et, de façon générale, $I_C(A, B) + I_C(B, A) = 1$. Cet indicateur permet d'obtenir une idée de la qualité d'une approximation par rapport à une autre en termes de convergence en un temps de calcul raisonnable.

1.3.1.5 Ensemble et point de référence

L'indicateur de contribution a l'avantage de ne pas nécessiter de paramètres. Au contraire, les indicateurs epsilon et hypervolume requièrent tout deux un ensemble de référence R, ce dernier nécessitant également la spécification d'un point de référence z^{ref}.
Idéalement, il semble évident de définir l'ensemble de référence R comme étant l'ensemble Pareto optimal X_E. Cependant, nous ne sommes pas en mesure de fournir cet ensemble pour la totalité des instances des deux problèmes abordés dans ce manuscrit. En conséquent, l'ensemble de référence associé à une instance de problème donnée sera calculé de la façon suivante. Soit $A^\star$ l'union de toutes les approximations obtenues lors de nos expérimentations pour une instance donnée. $A^\star$ contient probablement à la fois des solutions non-dominées et dominées, car une approximation peut contenir des éléments dominant ceux d'une autre approximation, et vice versa. L'ensemble de référence R est alors composé de l'ensemble des solutions non-dominées extraites de $A^\star$. Il correspond en d'autres mots au meilleur ensemble trouvé.
Soient $z^{min} = (z^{min}_1, \ldots, z^{min}_n)$ et $z^{max} = (z^{max}_1, \ldots, z^{max}_n)$, tels que z^{min}_i (respectivement z^{max}_i) dénote la borne inférieure (respectivement supérieure) des valeurs de la fonction objectif f_i parmi tous les points contenus dans $f(A^\star)$. Le vecteur z^{min} peut donc être vu comme une approximation du point idéal $z^\star$. Afin de donner une échelle à peu près équivalente à toutes les fonctions objectif, les valeurs objectif de toutes les approximations sont préalablement normalisées par rapport à

z^{min} et z^{max}. Par ailleurs, le point de référence z^{ref}, nécessaire au calcul de l'hypervolume, est fixé à z^{max}.

1.3.2 Validation statistique

Jusqu'ici, nous avons supposé que chacun des algorithmes considérés ne générait qu'une seule approximation pour une instance donnée. Néanmoins, les méthodes d'optimisation considérées ici sont des algorithmes stochastiques. À chaque exécution d'un tel algorithme, une approximation différente peut être retournée. Ainsi, afin d'analyser empiriquement la performance de nos algorithmes stochastiques, il faut commencer par exécuter plusieurs fois le même algorithme sur une même instance de problème. On obtient donc un échantillon d'approximations. Comparer deux méthodes revient donc à comparer les deux échantillons correspondant, ce qui conduit à utiliser un test d'hypothèse statistique. La démarche que nous allons suivre ici consiste à transformer les échantillons d'approximations obtenus en échantillon de I-valeurs scalaires. Des procédures de tests statistiques classiques sont ensuite appliquées sur ces échantillons de I-valeurs.
Considérons deux algorithmes A et B ayant été exécutés r fois chacun pour résoudre une même instance de problème. On obtient donc deux échantillons d'approximations correspondants : $(A_1, A_2, \ldots, A_r)$ et $(B_1, B_2, \ldots, B_r)$. Ensuite, relativement à un indicateur unaire [2] de qualité I, on transforme ces deux échantillons en deux autres échantillons dont les éléments sont dorénavant des valeurs scalaires : $(I(A_1), I(A_2), \ldots, I(A_r))$ et $(I(B_1), I(B_2), \ldots, I(B_r))$. Ces résultats peuvent être résumés à l'aide de statistiques descriptives telles que la moyenne de ces I-valeurs par algorithme. On peut également appliquer une procédure de test statistique standard afin d'être en mesure de déclarer que « *l'algorithme A surpasse l'algorithme B* », « *l'algorithme B surpasse l'algorithme A* », ou « *il n'existe pas de différence significative entre les algorithmes A et B* », ceci selon l'indicateur de qualité I spécifié (et pour une instance donnée). En effet, la situation peut être différente pour un autre indicateur ; c'est la raison pour laquelle plusieurs indicateurs sont généralement considérés.
Les tests non-paramétriques sont couramment utilisés pour comparer ce type d'échantillon de petite taille, car ils ne font aucune hypothèse sur la distributions des variables (Zitzler *et al.*, 2008). Deux cas peuvent alors se présenter lors de la comparaison de deux algorithmes A et B : soit chaque exécution résulte en un échantillon aléatoire complètement indépendant (échantillons indépendants), soit l'influence d'une ou plusieurs variables aléatoires est partiellement écartée (échantillons appariés). Dans notre cas, la population initiale ainsi que la graine du générateur de nombres pseudo-aléatoires sont identiques pour les exécutions correspondantes des deux algorithmes, les échantillons sont donc appariés. De ce fait, un test des rangs signés de Wilcoxon sera utilisé avec un niveau de signification $\alpha = 0.05$. Le nombre d'exécutions d'un algorithme par instance est fixé à $r = 20$.

1.4 Problèmes d'optimisation combinatoire multiobjectif, deux cadres applicatifs

Tout comme les problèmes d'optimisation monoobjectif, les problèmes d'optimisation multiobjectif peuvent être classés en deux catégories : ceux dont les solutions sont représentées à l'aide

2. Si un indicateur binaire est considéré, il suffit de le transformer en indicateur unaire à l'aide d'un ensemble de référence, comme en (1.4).

de variables réelles, comme les problèmes d'optimisation multiobjectif continus, et ceux dont les variables sont représentées à l'aide de variables discrètes, tels que les problèmes d'optimisation multiobjectif combinatoires. La plupart des travaux existants traitent de problèmes continus. Néanmoins, l'intérêt porté aux problèmes combinatoires s'accroît depuis plusieurs années. En effet, de nombreux problèmes réels et académiques classiques ont été modélisés en problèmes d'optimisation multiobjectif combinatoires.
Ici, nous allons nous intéresser à deux problèmes combinatoires académiques qui seront traités lors des analyses expérimentales menées dans les chapitres suivants. Le premier problème est le problème d'ordonnancement de type Flowshop de permutation déjà traité au sein d'un nombre relativement important de travaux, notamment au sein de l'équipe de recherche DOLPHIN ; voir par exemple la thèse de Basseur (2005). Le deuxième, le problème de Ring-Star, est une généralisation d'un problème de tournées monoobjectif qui est ici, et pour la première fois, formulé en un problème biobjectif.

1.4.1 Le problème de Flowshop

Le problème d'ordonnancement de type Flowshop (*Flowshop scheduling problem, FSP*) est l'un des problèmes d'ordonnancement les plus étudiés de la littérature. La plupart des travaux le considère comme un problème monoobjectif et vise principalement à minimiser le *makespan*, c'est-à-dire la date d'achèvement de l'ordonnancement. Pourtant, suivant les particularités du problème à traiter, un grand nombre de fonctions objectif peuvent être considérées, et différentes approches multiobjectif ont également été proposées (T'Kindt et Billaut, 2002; Landa Silva *et al.*, 2004).
Dans un premier temps, nous allons rapidement présenter les différentes classes de problèmes d'ordonnancement rencontrés dans le domaine de l'optimisation. Ensuite, une définition du problème traité sera donnée. Elle sera suivie par une description des jeux de données que nous utiliserons et par une brève revue des méthodes existantes pour la résolution de ce type de problème.

1.4.1.1 Introduction aux problèmes d'ordonnancement

Un problème d'ordonnancement se compose de tâches à réaliser et de machines disponibles pour les exécuter. La résolution d'un tel problème nécessite donc de déterminer l'ordre dans lequel chaque machine exécute les tâches ainsi que le déroulement des tâches au cours du temps. D'après la notation introduite par Graham *et al.* (1979), trois champs peuvent servir à décrire un problème d'ordonnancement : la structure, les contraintes et les objectifs du problème à résoudre.
Ainsi, la structure d'un problème d'ordonnancement peut prendre différentes formes. Voici quelques exemples.
- Un atelier à *machine unique*, où une seule machine est disponible.
- Un atelier de type *flowshop*, pour lequel l'utilisation de plusieurs machines s'effectue dans le même ordre pour toutes les tâches.
- Un atelier de type *jobshop*, pour lequel chaque tâche possède un ordre de passage sur les différentes machines qui lui est propre.
- Un atelier de type *openshop*, où les tâches peuvent être exécutées dans un ordre quelconque sur les différentes machines.

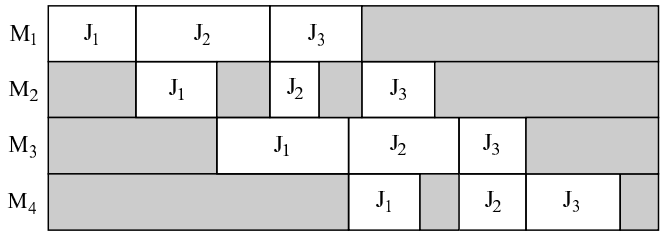


Fig. 1.15 – Exemple d'une solution d'un problème de Flowshop de permutation où trois jobs (J_1, J_2, J_3) doivent être ordonnancés sur quatre machines (M_1, M_2, M_3, M_4).

Les contraintes devant être respectées par un problème d'ordonnancement concernent, par exemple, des dates de disponibilités ou des dates dues pour chacun des jobs, ou peuvent encore prendre la forme de contraintes de précédence entre les jobs. Enfin, les fonctions objectif les plus classiques concernent probablement la date de fin de l'ensemble des travaux à ordonnancer (makespan), le temps de présence maximum d'une tâche sur l'atelier, ou encore le retard total ou maximum d'un job. Une description très complète des différentes valeurs possibles pour chacun des champs de structure, contraintes et objectifs est donnée par T'Kindt et Billaut (2002).

1.4.1.2 Définition du problème

Le problème de Flowshop consiste à ordonnancer un ensemble de N jobs $\{J_1, J_2, \ldots, J_N\}$ sur M machines $\{M_1, M_2, \ldots, M_M\}$. Les machines sont des ressources critiques, c'est-à-dire que deux jobs ne peuvent être affectés à une même machine simultanément. Un job J_i est composé de M tâches consécutives $\{t_{i1}, t_{i2}, \ldots, t_{iM}\}$, où t_{ij} représente la $j$ème tâche du job J_i et requiert donc la machine M_j. À chaque tâche t_{ij} est associée une durée d'exécution p_{ij}. Et à chaque job J_i est associée une date due d_i (la date de fin souhaitée du job). Comme illustré sur la figure 1.15, nous nous intéressons ici au problème de Flowshop de permutation, où la séquence des jobs est identique et unidirectionnelle sur chacune des machines. Ainsi, les solutions réalisables peuvent être représentées à l'aide de permutations. Pour une instance de taille N, il existe donc $N!$ solutions réalisables.

En fonction des particularités du problème traité, de nombreux critères peuvent être envisagés lors de l'ordonnancement de tâches sur plusieurs machines. Ici, nous allons nous intéresser à deux variantes du même problème.

- Un problème de Flowshop à deux fonctions objectif, que nous noterons FSP-2, et qui consiste à minimiser le makespan C_{max} (la date d'achèvement de l'ordonnancement) et la somme des retards $\overline{T}$.
- Une variante à trois fonctions objectif, FSP-3, où le retard maximum T_{max} est un objectif supplémentaire à minimiser.

Ces trois critères sont parmi les plus étudiés de la littérature (T'Kindt et Billaut, 2002; Landa Silva *et al.*, 2004). Soit C_{ij} la date de complétion de la tâche t_{ij}, les fonctions objectif peuvent être définies comme suit.

$$C_{max} = \max_{i \in \{1,\ldots,N\}} \{C_{iM}\} \tag{1.6}$$

$$\overline{T} = \sum_{i=1}^{N} \Big\{ \max\{0, C_{iM} - d_i\} \Big\} \tag{1.7}$$

$$T_{max} = \max_{i \in \{1,\ldots,N\}} \Big\{ \max\{0, C_{iM} - d_i\} \Big\} \tag{1.8}$$

D'après la notation définie par Graham *et al.* (1979), FSP-2 est de type $F/perm, d_i/(C_{max}, \overline{T})$, et FSP-3 est de type $F/perm, d_i/(C_{max}, \overline{T}, T_{max})$.

Minimiser le makespan, la somme des retards ou le retard maximum de façon indépendante est déjà NP-difficile pour des problèmes à plus de deux machines. Ainsi, les problèmes FSP-2 et FSP-3 étudiés ici le sont également. En conséquence, les instances de grande taille ne peuvent généralement pas être résolues de façon exacte.

1.4.1.3 Jeux de données

Afin d'évaluer les performances des approches proposées au cours de cette thèse, nous allons considérer différents jeux de données (Liefooghe *et al.*, 2007a). Ceux-ci ont été construits à partir des instances proposée par Taillard (1993) pour le problème de Flowshop monoobjectif. Ils ont été étendus au cas multiobjectif en y ajoutant une date due à chacun des jobs[3]. Les durées d'exécution des tâches sur les machines ont été générées aléatoirement, selon une distribution uniforme, dans l'intervalle $[0, 99]$. Ensuite, les dates dues ont été fixées à l'aide d'une valeur aléatoire définie entre $\overline{p} \times M$ et $\overline{p} \times (N + M - 1)$, où N représente le nombre de jobs, M le nombre de machines, et $\overline{p}$ le temps d'exécution moyen observé sur l'instance considérée. Ainsi, une date due se situe grossièrement entre la date de complétion moyenne du premier job ordonnancé et la date de complétion moyenne du dernier job ordonnancé. Des jeux de données sont proposés pour des problèmes de $N = \{20, 30, 50, 70, 100, 150, 200, 300, 500\}$ jobs et de $M = \{5, 10, 15, 20\}$ machines ; un ensemble de 10 instances différentes étant proposé par taille $(N \times M)$. Notez qu'une instance nommée $(N \times M \times i)$ se réfère à la $i$ème instance composée de N jobs et M machines.

1.4.1.4 État de l'art succinct des méthodes de résolution existantes

Les méthodes proposées dans la littérature pour résoudre les problèmes d'ordonnancement multiobjectif varient des méthodes exactes aux métaheuristiques en passant par les heuristiques dédiées. Dans leur état de l'art, Nagar *et al.* (1995) classent les problèmes d'ordonnancement selon différentes caractéristiques, notamment la configuration des ateliers (machine unique ou machines multiples) et des fonctions objectif (deux ou plus de deux objectifs). La majorité des travaux concernant le problème du Flowshop à machines multiples s'est restreinte à l'étude d'un seul objectif (généralement la date de fin d'ordonnancement ou la somme des temps de complétion). Par ailleurs, la majorité des travaux portant sur l'ordonnancement multiobjectif traite de problèmes à deux machines ou à deux objectifs, et se concentre presque exclusivement sur le problème de Flowshop. T'Kindt et Billaut (2002) fournissent une vue d'ensemble de l'ordonnancement multiobjectif destinée à la fois aux chercheurs et aux industriels. Ils fournissent des modèles et une topologie des problèmes d'ordonnancement monoobjectif et multiobjectif, et décrivent un ensemble de méthodes exactes et approchées pour les résoudre. Plus récemment, Landa Silva *et al.* (2004) ont réalisé une étude sur l'utilisation des métaheuristiques pour la résolution de problèmes d'ordonnancement multiobjectif. Les algorithmes évolutionnaires et les algorithmes de recherche locale semblent être les approches les plus représentées, tout comme les méthodes hybridant ces deux types de métaheuristique. Plus récemment, Minella *et al.* (2008) ont

3. Ces instances sont disponibles à l'URL : `http://www.lifl.fr/~liefooga/benchmarks/`.

fourni une revue de la littérature pour le problème de Flowshop de permutation multi-objectif, ainsi qu'une analyse expérimentale d'un certain nombre d'algorithmes sur plusieurs variantes biobjectif du problème étudié ici.

1.4.2 Le problème de Ring-Star

Le but du problème de Ring-Star (*ring star problem*) consiste à localiser un cycle élémentaire (un anneau) dans un sous-ensemble de nœuds d'un graphe, tout en optimisant deux types de coût contradictoires. La première fonction objectif consiste à minimiser un coût-anneau proportionnel à la longueur du cycle. La deuxième fonction objectif consiste à minimiser un coût d'affectation des nœuds non visités vers les nœuds visités du graphe. En dépit de sa formulation naturelle en problème biobjectif, celui-ci est généralement traité de façon monoobjectif (Labbé *et al.*, 2004, 2005). Cependant, d'un point de vue pratique, il est souvent nécessaire de prendre différents objectifs en compte simultanément, et le problème étudié ici est une parfaite illustration de ce qui peut survenir dans le monde industriel. En effet, comme l'ont souligné Jozefowiez *et al.* (2008), un grand nombre de problèmes de tournées peuvent être formulés comme des problèmes d'optimisation multiobjectif, et le problème étudié ici est une généralisation des problèmes monoobjectif introduits par Labbé *et al.* (2004, 2005). Par ailleurs, le problème de Ring-Star est fortement combinatoire, car une fois qu'est décidé quels nœuds sont visités, un problème de voyageur de commerce classique (*traveling salesman problem*, TSP) reste à résoudre. Par conséquent, les instances de grande taille ne peuvent généralement pas être résolues de façon exacte. C'est la raison pour laquelle des méthodes de recherche heuristiques sont considérées dans le présent document.

Après une brève présentation des problèmes de tournées multiobjectif, nous allons formuler le problème de Ring-Star étudié ici, et présenter les jeux de données que nous allons utiliser. Ensuite, une revue succincte des méthodes de résolution existantes et quelques intérêts industriels liés à ce problème seront discutés.

1.4.2.1 Introduction aux problèmes de tournées

Les problèmes de tournées, tels que le problème du voyageur de commerce ou le problème de tournées de véhicules, sont fréquemment étudiés du fait de leurs nombreuses applications réelles, notamment en transport et logistique. Les objectifs les plus courants consistent à minimiser la distance parcourue, le temps requis, le coût de la tournée, la taille de la flotte de véhicules, ou encore à maximiser la qualité de service ou le profit collecté. D'autres aspects, tels que l'équilibrage des tournées, peuvent également être pris en compte. D'après Jozefowiez *et al.* (2008), les problèmes de tournées multiobjectif se présentent généralement sous trois formes : l'extension de problèmes académiques existants dans le but d'améliorer leur application pratique, la généralisation de problèmes classiques, ou l'étude de cas réels où différents objectifs ont clairement été identifiés.

Le problème de Ring-Star étudié ici correspond clairement à une généralisation du problème monoobjectif original, pour lequel soit les deux fonctions objectif étaient combinées (Labbé *et al.*, 2004), soit l'une des deux fonctions objectif était traitée comme une contrainte (Labbé *et al.*, 2005).

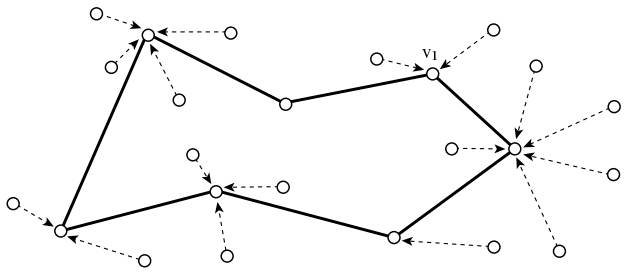


Fig. 1.16 – Un exemple de solution par le problème de Ring-Star.

1.4.2.2 Définition du problème

Le problème de Ring-Star (RSP) peut être défini comme suit. Soit $G = (V, E, A)$ un graphe mixte complet où $V = \{v_1, v_2, \ldots, v_n\}$ est un ensemble de sommets, $E = \{[v_i, v_j] | v_i, v_j \in V, i < j\}$ est un ensemble d'arêtes, et $A = \{(v_i, v_j) | v_i, v_j \in V\}$ est un ensemble d'arcs. Le sommet v_1 représente le dépôt. À chaque arête $[v_i, v_j] \in E$ est associé un coût-anneau (*ring cost*) non nul, noté c_{ij}, et à chaque arc $(v_i, v_j) \in A$ est associé un coût d'affectation (*assignment cost*) non nul, noté d_{ij}. Le problème de Ring-Star biobjectif consiste à identifier un cycle sur un sous-ensemble de nœuds du graphe $V' \subset V$ (avec $v_1 \in V'$) tout en (i) minimisant la somme des coûts-anneau des arêtes du cycle, et en (ii) minimisant la somme des coûts d'affectation des arcs allant de chaque nœud non visité vers un nœud visité (de sorte que le coût d'affectation associé soit minimum). Un exemple d'une solution réalisable est donné sur la figure 1.16, où les lignes pleines représentent les arêtes appartenant à l'anneau, et où les lignes en pointillé représentent les arcs des affections entre les nœuds.
La première fonction objectif, le coût-anneau, est définie comme suit.

$$\sum_{[v_i, v_j] \in E} c_{ij} b_{ij} \tag{1.9}$$

où b_{ij} est une variable binaire égale à 1 si et seulement si l'arête $[v_i, v_j]$ appartient au cycle. Le second objectif, le coût d'affectation, est défini ainsi :

$$\sum_{v_i \in V \setminus V'} \min_{v_j \in V'} d_{ij} \tag{1.10}$$

Remarquons que ces fonctions objectif ne sont comparables entre elles que si nous supposons que le coût du cycle et les coûts d'affectation sont proportionnels l'un à l'autre, ce qui est rarement le cas en pratique. De plus, privilégier un coût par rapport à l'autre s'apparente fortement aux préférences du décideur. Or, nous nous plaçons ici dans un contexte de prise de décision *a posteriori*. Le problème de Ring-Star biobjectif est un problème combinatoire NP-difficile, car le cas particulier consistant à visiter l'ensemble des nœuds du graphe ($V' = V$) équivaut à un problème du voyageur de commerce traditionnel.

1.4.2.3 Jeux de données

Les expérimentations que nous allons réaliser au cours des chapitres suivants seront conduites sur un ensemble de jeux de données provenant de la TSPLIB [4] (Reinelt, 1991). Initialement

4. `http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/`.

créées pour le problème du voyageur de commerce, ces instances s'appliquent parfaitement à notre cas, et sont d'ailleurs utilisées au sein des travaux traitant d'une version à objectif unique du problème (Labbé *et al.*, 2004, 2005). Soit l_{ij} la distance entre deux nœuds v_i et v_j fournie par une instance de la TSPLIB. Pour chaque paire de nœuds v_i, v_j, le coût-anneau c_{ij} et le coût d'affectation d_{ij} sont tous deux fixés à l_{ij}.

1.4.2.4 État de l'art succinct des méthodes de résolution existantes

Le problème de Ring-Star appartient à la classe des problèmes de « location-allocation » visant à localiser des structures au sein d'un graphe (Labbé *et al.*, 1998). Il a été initialement formulé par Labbé *et al.* (1999) sous deux variantes. Sous sa première formulation (noté MCP1 par Labbé *et al.* (1999) et généralement appelé « *Ring-Star problem* »), une somme pondérée des deux fonctions objectif est à optimiser. Sous sa deuxième formulation (MCP2 (Labbé *et al.*, 1999), habituellement nommé « *median cycle problem* »), le coût de l'anneau est à minimiser tandis que le coût d'affectation doit respecter une borne donnée. Même si cela n'était pas explicitement mis en évidence par les auteurs originaux, ces deux formulations sont couramment utilisées pour convertir un problème d'optimisation multiobjectif en un problème monoobjectif à l'aide d'approches scalaires. En effet, elles correspondent respectivement à une transformation par méthode d'agrégation et par méthode ϵ-contrainte dans la littérature traitant de l'optimisation multiobjectif (Miettinen, 1999), voir la section 2.1.3.1.
La première formulation du problème a été largement étudiée par Labbé *et al.* (2004). Les auteurs proposent une méthode de type *branch-and-bound* et résolvent des instances de la TSPLIB et des instances générées aléatoirement allant jusqu'à deux cents nœuds en moins de deux heures. Ensuite, Labbé *et al.* (2005) ont résolu la seconde formulation du problème à l'aide d'une méthode similaire. Pour finir, l'une ou les deux versions du problème ont été attaquées à l'aide d'heuristiques telles qu'une recherche tabou à voisinage variable (Moreno Pérez *et al.*, 2003), un algorithme évolutionnaire (Renaud *et al.*, 2004), une heuristique gloutonne itérée (Renaud *et al.*, 2004), et une recherche tabou à voisinage variable hybridée à une recherche gloutonne adaptative (Dias *et al.*, 2006).
Un problème très similaire, à savoir le problème de Ring-Star à capacité a été introduit par Baldacci *et al.* (2007). Il étend la première formulation du problème de Ring-Star en limitant le nombre de nœuds pouvant être assignés à l'anneau (la capacité d'anneau). Ce problème a été résolu par une méthode exacte de type *branch-and-bound*. Notez que le même problème a également été abordé par Mauttone *et al.* (2007) à l'aide d'une métaheuristique hybride. En outre, Beasley et Nascimento (1996) ont présenté un problème de tournées-allocation à véhicule unique (*single vehicle routing-allocation problem*, SVRAP) dans lequel les nœuds non visités peuvent soit être affectés au cycle, soit isolés. Dans ce cas, un objectif additionnel minimisant le nombre de nœuds isolés doit être pris en compte. Vogt *et al.* (2007) ont proposé une recherche tabou pour résoudre un problème monoobjectif résultant de cette formulation.
Comme le montre l'état de l'art proposé par Jozefowiez *et al.* (2008), le nombre de problèmes de tournées multiobjectif s'est accru durant les dernières années. Mais, malgré ses nombreuses applications industrielles (voir la section suivante), le problème de Ring-Star n'a jamais été étudié comme un problème multiobjectif. Néanmoins, il est à noter que Current et Schilling (1994) ont défini deux variantes d'un problème multiobjectif très similaires au problème de Ring-Star : le *median tour problem* et le *maximal covering tour problem*. Dans ces deux variantes, l'une des fonctions objectif consiste à minimiser la longueur totale du tour, alors que l'autre consiste à

maximiser l'accès au tour pour les nœuds non visités. Pour s'attaquer à ces deux problèmes, les auteurs utilisent une approche de type lexicographique, où une hiérarchie est définie entre les fonctions objectif, et où chaque fonction objectif est considérée l'une après l'autre. Par ailleurs, Doerner *et al.* (2007) ont récemment formulé un problème de planification de tournées pour les établissements de soins mobiles. Trois objectifs sont pris en compte et le problème résultant est relativement proche de celui introduit par Beasley et Nascimento (1996). Un établissement mobile doit visiter un sous-ensemble de nœuds. Les nœuds non visités sont affectés à l'arrêt du tour le plus proche, ou sont considérés comme ne pouvant pas accéder aux soins (par rapport à une distance maximale donnée). Les objectifs étudiés sont (*i*) la minimisation du ratio entre le temps non productif du personnel médical et le temps total, (*ii*) la minimisation de la distance moyenne à l'arrêt le plus proche, et (*iii*) la maximisation d'un critère de couverture. Les auteurs utilisent un algorithme de colonie de fourmis multiobjectif basé sur la dominance Pareto, ainsi que deux algorithmes évolutionnaires multiobjectif classiques, à savoir VEGA (Schaffer, 1985) et MOGA (Fonseca et Fleming, 1993), pour résoudre un cas concret du problème.

1.4.2.5 Intérêts industriels

Le problème de Ring-Star possède un large éventail d'intérêts industriels, notamment dans le domaine des télécommunications et des problèmes de tournées. Ainsi, une telle formulation peut facilement être appliquée à la conception de systèmes de prestation de services mobiles (prestation des soins de santé dans les zones rurales des pays en voie de développement), de systèmes de transport bimodal (par exemple, la distribution du courrier de nuit), ou encore de réseaux informatiques distribués. Bien sûr, il est évident que les applications pratiques, issues du monde réel, posséderaient très probablement un ensemble de contraintes supplémentaires.

Par exemple, comme indiqué par Labbé *et al.* (2004, 2005), un réseau en anneau doit être conçu afin d'interconnecter un ensemble de hubs dans le cas de services de données numériques (Xu *et al.*, 1999). Des concentrateurs doivent être installés sur un sous-ensemble de lieux et doivent être interconnectés sur un réseau en anneau (l'Internet), tandis que les emplacements restants sont affectés à ces concentrateurs (l'Intranet). Des problèmes relativement proches surviennent lors de la planification des systèmes de transport en commun ou lors de la conception de réseaux optiques. De plus, d'autres applications existent dans la collecte postale, ou encore dans la conception d'itinéraires de livraison, où la distance entre un client et un point de collecte se doit d'être raisonnable. Ainsi, le positionnement de boîtes postales prenant en compte à la fois les coûts engendrés par la collecte et la gène occasionnée pour les utilisateurs a été étudié par Labbé et Laporte (1986). D'autres applications étroitement liées au problème de Ring-Star concernent la conception d'infrastructures de transport de forme circulaire (comme les lignes de métro ou les autoroutes), l'emplacement des bacs de collecte des déchets recyclables, et l'itinéraire d'autobus scolaires. Enfin, l'acheminement de services de santé, étudié par de nombreux auteurs parmi lesquels Akinc et Srikanth (1992) ainsi que Doerner *et al.* (2007), se compose d'une clinique mobile desservant un secteur sans être en mesure de visiter tous les nœuds de la population locale. Ainsi, les nœuds non visités doivent se rendre à l'arrêt de la tournée le plus proche par leurs propres moyens pour être traités médicalement.

Conclusion

Dans ce chapitre, nous avons traité différents aspects de l'aide à la décision multicritère en général, et de l'optimisation multiobjectif en particulier. Nous avons abordé les questions fondamentales à prendre en compte lors de la résolution d'un problème d'optimisation multiobjectif. Elles sont résumées ci-dessous.

Définitions. La formulation générale d'un problème d'optimisation multiobjectif a été définie, ainsi que les notions d'optimalité, les relations de dominance Pareto, de dominance faible, de dominance stricte, d'ϵ-dominance, et les bornes du front Pareto (point idéal, point nadir). Diverses notations liées à ces différentes questions ont par ailleurs été introduites.

Approches de résolution. Les approches de résolution de problèmes d'optimisation multiobjectif ont été discutées, que ce soit vis-à-vis de l'intervention du décideur au sein du processus de décision, ou au sujet des méthodologies de résolution à utiliser, en particulier les méthodes de type métaheuristique. Nous avons également précisé que les études menées au cours des chapitres suivants porteront essentiellement sur une approche *a posteriori*, qui visent à identifier l'ensemble Pareto optimal (ou une bonne approximation de celui-ci).

Analyse de performances. L'importance de l'analyse de la performance et de la qualité des méthodes d'optimisation multiobjectif a été mise en avant, notamment à l'aide d'indicateurs de qualité (en particulier les indicateurs hypervolume et epsilon, complémentaires l'un de l'autre) et d'une validation statistique des résultats obtenus.

Ensuite, nous avons présenté deux problèmes applicatifs, rencontrés dans le domaine de la logistique, sur lesquels portera l'analyse expérimentale des chapitres suivants.

Flowshop. Le problème de Flowshop que nous allons traiter est un problème d'optimisation combinatoire issu de l'ordonnancement multiobjectif. Par la suite, nous allons considérer deux variantes de ce même problème, à deux ou à trois objectifs. Pour cela, nous avons adapté une série de jeux de données monoobjectif classiques au cas multiobjectif.

Ring-Star. Le problème de Ring-Star est un problème de tournées qui possèdent de nombreuses applications industrielles. En dépit de son aspect multiobjectif évident, il est ici formulé comme un problème biobjectif pour la première fois. Cette formulation peut donc être vue comme une généralisation d'un problème monoobjectif existant.

Les connaissances apportées jusqu'ici représentent le fondement théorique et applicatif des contributions qui seront présentées lors du prochain chapitre pour la conception, l'implémentation et l'analyse expérimentale de métaheuristiques pour l'optimisation multiobjectif.

Chapitre 2

Métaheuristiques pour l'optimisation multiobjectif

Ce chapitre est consacré à la conception, l'implémentation et l'analyse expérimentale de métaheuristiques pour l'optimisation multiobjectif. Les principales contributions qui en résultent sont synthétisées ci-dessous.

- *Une vue unifiée de la conception de métaheuristiques pour l'optimisation multiobjectif, et son implémentation au sein d'une plateforme logicielle générique, ParadisEO-MOEO (Liefooghe* et al.*, 2009c).*
- *De nouvelles métaheuristiques pour l'optimisation multiobjectif : un algorithme évolutionnaire élitiste appelé SEEA (Liefooghe* et al.*, 2010), et divers algorithmes de recherche locale basés sur la dominance Pareto (Liefooghe* et al.*, 2009e).*
- *La modélisation et la résolution du problème de Ring-Star comme un problème d'optimisation multiobjectif (Liefooghe* et al.*, 2008b).*

Sommaire

Principales publications en rapport avec le chapitre

LIEFOOGHE, A., BASSEUR, M., JOURDAN, L. et TALBI, E.-G. (2007). ParadisEO-MOEO : A framework for evolutionary multi-objective optimization. *In Fourth International Conference on Evolutionary Multi-criterion Optimization (EMO 2007)*, volume 4403 de *Lecture Notes in Computer Science*, pages 386–400, Matsushima, Japan. Springer-Verlag.

LIEFOOGHE, A., JOURDAN, L., BASSEUR, M., TALBI, E.-G. et BURKE, E. K. (2008). Metaheuristics for the bi-objective ring star problem. *In Eighth European Conference on Evolutionary Computation in Combinatorial Optimisation (EvoCOP 2008)*, volume 4472 de *Lecture Notes in Computer Science*, pages 206–217, Napoli, Italy. Springer-Verlag.

LIEFOOGHE, A., JOURDAN, L., LEGRAND, T., HUMEAU, J. et TALBI, E.-G. (2009). ParadisEO-MOEO : A software framework for evolutionary multi-objective optimization. *In Advances in multi-objective nature inspired computing*, Studies in Computational Intelligence. Springer-Verlag. (à paraître).

LIEFOOGHE, A., JOURDAN, L. et TALBI, E.-G. (2009). A unified model for evolutionary multi-objective optimization and its implementation in a general purpose software framework. *In IEEE Symposium on Computational Intelligence in Multicriteria Decision-Making (IEEE MCDM 2009)*, pages 88–95, Nashville, Tennessee, USA. IEEE Press.

LIEFOOGHE, A., MESMOUDI, S., HUMEAU, J., JOURDAN, L. et TALBI, E.-G. (2009). A study on dominance-based local search approaches for multiobjective combinatorial optimization. *In Workshop on Engineering Stochastic Local Search Algorithms. Designing, Implementing and Analyzing Effective Heuristics (SLS 2009)*, volume 5752 de *Lecture Notes in Computer Science*, pages 120–124, Brussels, Belgium. Springer-Verlag.

Articles soumis

LIEFOOGHE, A., JOURDAN, L. et TALBI, E.-G. (2009). The ParadisEO-MOEO software framework : A unified design and implementation of evolutionary multiobjective optimization algorithms. *European Journal of Operational Research*. (soumis).

LIEFOOGHE, A., MESMOUDI, S., HUMEAU, J., JOURDAN, L. et TALBI, E.-G. (2009). On dominance-based local search : Design, implementation and experiments on multiobjective scheduling and traveling salesman problems. *Journal of Heuristics*. (soumis).

Introduction

L'intention du présent chapitre est multiple, et concerne la conception, l'implémentation, ainsi que l'analyse expérimentale de métaheuristiques pour l'optimisation multiobjectif. Dans un premier temps, une vision commune de la conception de métaheuristiques pour l'optimisation multiobjectif est donnée. Nous décrivons les composants fondamentaux partagés par toute métaheuristique, et nous introduisons une classification fine de ces composants. Cette vue unifiée est ensuite approfondie à l'aide d'une étude plus poussée de deux types de méthodologie particuliers. Deux cadres de conception généraux sont donc proposés ; l'un pour les algorithmes évolutionnaires, l'autre pour les algorithmes de recherche locale. Nous confirmons la généricité et la modularité élevée de la vue unifiée en traitant un certain nombre de méthodes classiques, et représentatives de l'état de l'art actuel, comme de simples instances des deux modèles présentés. Enfin, nous illustrons comment ces modèles nous ont permis de proposer de nouvelles approches de résolution générales pour l'optimisation multiobjectif.

Les algorithmes évolutionnaires se montrent particulièrement bien adaptés au traitement de problèmes multiobjectif, grâce à leur habilité à trouver un ensemble de solutions en une seule exécution. L'optimisation multiobjectif évolutionnaire constitue un domaine de recherche très développé, et pour lequel de multiples avancées, notamment algorithmiques, ont été réalisées au cours des dernières années. En effet, de nombreuses méthodologies de ce type ont été proposées dans la littérature. Elles se placent généralement à un haut niveau d'abstraction et de généricité du fait de leur fonctionnement indépendant du problème à résoudre. Ici, nous nous plaçons à un niveau d'abstraction encore plus élevé en considérant un ensemble représentatif de ces algorithmes évolutionnaires multiobjectif comme des spécifications différentes d'un seul et même modèle. Quant à eux, les algorithmes de recherche locale se font plutôt rares dans la littérature. Ils se basent généralement sur l'amélioration itérée d'une solution unique, et doivent dès lors être adaptés aux besoins de l'optimisation multiobjectif, et donc à la recherche d'un ensemble de solutions. Le travail présenté ici est en ce sens plus novateur d'un point de vue algorithmique.

Dans un second temps, nous illustrons la façon dont ce modèle polyvalent a été utilisé comme point de départ pour la conception et la mise en œuvre d'une plateforme logicielle open-source dédiée à l'implémentation de métaheuristiques pour l'optimisation multiobjectif : ParadisEO-MOEO [1]. Tous les choix d'implémentation ont été fortement motivés par la vue unifiée présentée au cours de cette thèse. Cette plateforme logicielle a été largement expérimentée et a permis la résolution d'une grande diversité de problèmes, à la fois académiques et réels, issue de l'optimisation multiobjectif continue ou combinatoire. En comparaison à l'existant, nous pensons que le modèle unifié proposé ici est plus complet, qu'il fournit une décomposition plus fine, et que la plateforme logicielle ParadisEO-MOEO offre une implémentation plus modulaire que les tentatives qui ont été précédemment faites en ce sens.

Dans un dernier temps, nous étudions le comportement de diverses métaheuristiques, à la fois classiques et novatrices, pour la résolution des deux applications qui nous intéressent ici : les problèmes de Flowshop et de Ring-Star. Tout d'abord, une modélisation pour chacun de ces problèmes est proposée. Ensuite, une analyse expérimentale rigoureuse est menée sur un nombre d'approches adaptées au problème traité. Les expérimentations que nous avons réalisées montrent que les algorithmes étudiés sont facilement extensibles, et qu'ils parviennent à trouver une approximation de bonne qualité de l'ensemble Pareto optimal, ceci pour des problèmes de types

1. La plateforme ParadisEO-MOEO est disponible à l'URL : `http://paradiseo.gforge.inria.fr`.

différents et de tailles différentes. Nous discutons de leurs comportements respectifs et nous essayons d'émettre quelques lignes directrices à propos des principaux composants de conception liés à la résolution de problèmes d'optimisation combinatoire multiobjectif.

Le chapitre est organisé de la façon suivante. La section suivante présente les questions fondamentales liées à la conception de métaheuristiques pour l'optimisation multiobjectif en général, et d'algorithmes évolutionnaires et de méthodes de recherche locale en particulier. La section 2.2 est dédiée aux aspects d'implémentation, et à la plateforme logicielle ParadisEO-MOEO. Enfin, les méthodes de résolution proposées pour les problèmes de Flowshop et de Ring-Star, ainsi qu'une analyse expérimentale des métaheuristiques étudiées font l'objet de la section 2.3.

2.1 Conception

2.1.1 Motivations

Cette section tente de soulever les questions fondamentales liées à la conception de métaheuristiques pour l'optimisation multiobjectif. L'application de métaheuristiques pour la résolution de problèmes d'optimisation multiobjectif est devenue un domaine de recherche et de développement très actif au cours des dernières années. Résoudre cette classe de problèmes consiste à identifier un ensemble de solutions de bonne qualité, répondant aux conditions de convergence vers le front Pareto et de diversité uniforme. Mais produire un tel ensemble est généralement impossible, en raison de la complexité du problème sous-jacent ou du grand nombre d'optima. Par conséquent, il convient fréquemment d'identifier une bonne approximation de celui-ci. Ainsi, la plupart des travaux consacrés aux métaheuristiques pour l'optimisation multiobjectif se concentrent sur les métaheuristiques à base de population, car ces dernières permettent tout naturellement de trouver de multiples solutions en une seule exécution. Plus particulièrement, un très grand nombre d'algorithmes évolutionnaires multiobjectif ont été proposés ; et des avances majeures, à la fois théoriques et algorithmiques, ont été accomplies dans le champ de l'optimisation évolutionnaire multiobjectif (Deb, 2001; Coello Coello *et al.*, 2007). Parmi les approches de résolution existantes, on peut par exemple citer les algorithmes VEGA (Schaffer, 1985), MOGA (Fonseca et Fleming, 1993), NSGA (Srinivas et Deb, 1994), NSGA-II (Deb *et al.*, 2002), NPGA (Horn *et al.*, 1994), SPEA (Zitzler et Thiele, 1999), SPEA2 (Zitzler *et al.*, 2001b) et PESA (Corne *et al.*, 2000).

Métaheuristiques. Notre attention se tourne principalement vers le développement de métaheuristiques permettant de trouver une approximation de l'ensemble Pareto optimal d'un problème d'optimisation multiobjectif. Une vue unifiée pour la conception de métaheuristiques dédiées à l'optimisation multiobjectif est proposée. L'accent est particulièrement posé sur les composants de recherche requis pour l'adaptation de métaheuristiques en vue de résoudre des problèmes multiobjectif. Notez que l'aspect d'aide à la décision, portant sur le choix de la solution finale parmi les solutions non-dominées, n'est pas traité. Nous référons le lecteur au livre de Miettinen (1999) pour obtenir des éléments de réponse liés à l'aide à la décision multicritère. Ainsi, le modèle de conception unifié proposé peut être vu comme une méthodologie générale possédant un plus haut niveau d'abstraction que les algorithmes proposés jusqu'à présent. En effet, la formulation d'un modèle de conception générique est une pratique pertinente afin d'abstraire la conception et l'implémentation de détails spécifiques, et de fournir une formulation générale.

Jusqu'à présent, chaque métaheuristique pour l'optimisation multiobjectif était conçue indépendamment des autres, et implémentée comme une méthode autonome possédant ses propres éléments spécifiques. Par la suite, nous identifions les principaux composants de recherche partagés par toutes les métaheuristiques pour l'optimisation multiobjectif. Un modèle d'unification essaye de fournir une terminologie et une classification communes. Une telle classification générale donne une description et une comparaison communes aux algorithmes, et permettra la conception de nouvelles méthodes empruntant des idées aux algorithmes existants ou en combinant différentes stratégies de façon novatrice. En particulier, nous allons nous intéresser aux algorithmes évolutionnaires multiobjectif, pour lesquels il existe une quantité importante de travaux, et à une classe d'algorithmes de recherche locale multiobjectif pour lesquels le niveau de maturité est beaucoup moins avancé ; cette dernière contribution est donc en ce sens un peu plus novatrice d'un point de vue algorithmique.

Algorithmes évolutionnaires. Zitzler *et al.* (2004) remarquent que les approches évolutionnaires initiales portaient principalement sur le déplacement vers le front de Pareto (Schaffer, 1985; Fourman, 1985). Ensuite, des mécanismes de préservation de la diversité ont rapidement émergé (Fonseca et Fleming, 1993; Srinivas et Deb, 1994; Horn *et al.*, 1994). Puis, à la fin des années 1990, la notion d'élitisme, liée à la préservation et à l'utilisation des solutions non-dominées, est devenue très populaire et est employée dans les stratégies les plus récentes (Zitzler et Thiele, 1999; Knowles et Corne, 2000; Zitzler *et al.*, 2001b). Ainsi, les questions essentielles d'affectation des valeurs de fitness, de préservation de la diversité et d'élitisme sont communément admises au sein de la communauté. Sur la base de ces trois notions principales, plusieurs tentatives ont déjà été réalisées dans le passé pour unifier les algorithmes évolutionnaires multiobjectif. Laumanns *et al.* (2000) ont mis l'accent sur les méthodes de recherche élitistes. Cette étude a été ultérieurement étendue par Zitzler *et al.* (2004), où les concepts algorithmiques d'affectation des valeurs de fitness, de préservation de la diversité et d'élitisme sont largement discutés. Plus récemment, Deb (2008) a proposé un cadre « robuste » pour l'optimisation évolutionnaire multiobjectif basé sur NSGA-II. Cette dernière approche se décompose en trois composants principaux liés à la préservation des solutions élites, l'accentuation des solutions non-dominées, et le maintien de la diversité. Toutefois, ce modèle est strictement axé sur NSGA-II, alors que d'autres méthodes classiques peuvent tout aussi bien être décomposées de la même manière. En effet, beaucoup de composants sont partagés par de nombreux algorithmes évolutionnaires multiobjectif, de sorte que, en un sens, ils peuvent tous être considérés comme des variantes d'un même modèle unifié comme nous allons le voir par la suite.

Algorithmes de recherche locale. L'importance de l'optimisation évolutionnaire multiobjectif n'est plus à démontrer et ce domaine de recherche est arrivé à un degré de maturité avancé. D'un autre côté, les méthodes de recherche locale sont réputées pour fournir des solutions de bonne qualité à de nombreux problèmes difficiles de l'optimisation combinatoire monoobjectif. Cependant, en comparaison au nombre considérable d'algorithmes évolutionnaires multiobjectif existants dans la littérature, le nombre d'approches de recherche locale multiobjectif est extrêmement faible. Ces dernières offrent pourtant une alternative intéressante aux algorithmes évolutionnaires classiques, et manipulent souvent un nombre relativement réduit de paramètres. Cependant, ils sont généralement restreints à un rôle de sous-procédure dans le cadre d'une métaheuristique hybride. En outre, d'un point de vue pratique, il apparaît souvent qu'une méthode

de recherche locale monoobjectif bien rodée, possédant des composants de haut niveau pour le problème à résoudre, doit être étendue afin de résoudre une variante à plusieurs objectifs du même problème. Dans un tel cas, considérer la conception d'un algorithme évolutionnaire multiobjectif ne semble pas naturel, mais c'est pourtant la voie la plus couramment suivie. De ce fait, nous nous concentrerons également sur les méthodes de recherche locale proposées pour l'optimisation multiobjectif, et nous proposons un modèle de conception unifié pour ce type de méthode basé sur une relation de dominance. Bien sûr, afin de trouver un ensemble de solutions non-dominées, nous allons devoir étendre les algorithmes de recherche locale classiques, qui sont des métaheuristiques à base de solution unique, au cas où une population de solutions évoluant en parallèle serait considérée. Nous essayerons d'esquisser les principales questions à étudier avant d'adapter ou de concevoir une méthode de recherche locale basée sur une relation de dominance. Évidemment, le succès de telles méthodes de recherche locale multiobjectif lors de l'identification d'une bonne approximation de l'ensemble Pareto optimal est étroitement lié à la connexité entre les solutions optimales pour le problème à résoudre (Ehrgott et Klamroth, 1997), mais ceci n'est pas le sujet du présent travail.

2.1.2 Concepts communs à tout type de métaheuristique

Il existe deux questions fondamentales de conception qui sont liées à toutes métaheuristiques : la *représentation* des solutions manipulées par l'algorithme, et l'*évaluation* de ces solutions dans l'espace objectif.

2.1.2.1 Représentation

La représentation d'une solution est le point de départ de la conception de tout type de métaheuristique. Une solution doit être représentée aussi bien dans l'espace décisionnel que dans l'espace objectif. Bien que la représentation dans l'espace objectif puisse être considérée comme indépendante du problème à traiter (si l'on considère, sans perte de généralité, que l'ensemble des points réalisables de l'espace objectif $Z \subseteq \mathbb{R}^n$), la représentation dans l'espace décisionnel doit être pertinente et adaptée au problème. En effet, le choix de cette représentation joue un rôle majeur sur l'efficacité de n'importe quelle métaheuristique et constitue donc une étape de conception essentielle. Le succès de l'application de métaheuristiques est fortement dépendant de cette représentation. Par ailleurs, le choix d'une représentation aura une influence considérable sur la façon dont les solutions seront manipulées par les opérateurs de recherche et lors de l'étape d'évaluation. Différentes représentations peuvent exister pour un même problème. Aussi, de nombreuses représentations simples peuvent être appliquées à différentes familles de problèmes d'optimisation, telles que les représentations basées sur des vecteurs de valeurs réelles, discrètes, binaires, ou des permutations.

2.1.2.2 Évaluation

L'étape d'évaluation correspond au calcul et à l'affectation des valeurs objectif pour une solution donnée. Dans le cadre de l'optimisation monoobjectif, une fonction objectif unique formule le but de l'optimisation. À chaque solution réalisable est donc associée une seule valeur scalaire qui quantifie la qualité de la solution. Ainsi, il existe un ordre total entre les solutions de l'espace de recherche. Pour un problème multiobjectif, il existe maintenant un ensemble de fonctions

objectif. De ce fait, l'évaluation associe, à chaque solution réalisable, un vecteur de valeurs dont chaque élément quantifie la qualité de la solution considérée par rapport à la fonction objectif correspondante. Ainsi, il n'existe plus d'ordre total, mais uniquement un ordre partiel entre les solutions, donné grâce à une relation de dominance. L'étape d'évaluation est un élément important de la conception d'une métaheuristique. Elle permet de guider la méthode vers les bonnes solutions de l'espace de recherche. Notez qu'en pratique, pour des problèmes réels, l'évaluation est bien souvent l'étape la plus coûteuse de la méthode de résolution en termes de temps de calcul.

2.1.3 Concepts communs à tout type de métaheuristique pour l'optimisation multiobjectif

Comme souligné au chapitre précédent, trouver une approximation de l'ensemble Pareto optimal est en soi un problème biobjectif. En effet, cette approximation doit à la fois satisfaire de bonnes propriétés en termes de convergence et de diversité : son image dans l'espace objectif doit être (*i*) proche, et (*ii*) uniformément répartie le long du front Pareto (ou d'un sous-ensemble de celui-ci). En conséquence, les principales différences entre la conception d'une métaheuristique pour l'optimisation monoobjectif et d'une métaheuristique pour l'optimisation multiobjectif traitent de ces deux critères.
Ainsi, la conception de métaheuristiques pour l'optimisation multiobjectif se base sur les trois principaux composants de recherche suivants.

- **Affectation d'une valeur de fitness**. Le rôle de ce processus est de guider la recherche vers les solutions Pareto optimales pour une meilleure convergence. Une valeur de fitness scalaire est ainsi affectée à un vecteur de valeurs objectif.
- **Préservation de la diversité**. Afin de ne pas converger vers un sous-ensemble du front Pareto, une stratégie de préservation de la diversité doit être mise en place pour générer un ensemble diversifié de solutions non-dominées.
- **Élitisme**. La notion d'élitisme a pour but de préserver et d'utiliser les solutions élites, c'est à dire les solutions non-dominées trouvées par l'algorithme.

Les concepts d'affectation de la fitness, de préservation de la diversité et d'élitisme sont communément approuvés par la communauté et sont présentés, par exemple, par Zitzler *et al.* (2004) ainsi que Coello Coello *et al.* (2007).

2.1.3.1 Affectation d'une valeur de fitness

Dans le cas monoobjectif, la valeur de fitness affectée à une solution est généralement sa valeur objectif unidimensionnelle. Lors de la résolution d'un problème d'optimisation multiobjectif, la valeur de fitness d'une solution vise à orienter la recherche vers les solutions Pareto optimales pour une meilleure convergence. Nous proposons de classer les stratégies d'affectation d'une valeur de fitness existantes en quatre familles (Fig. 2.1) : les approches scalaires, les approches basées sur un critère, les approches basées sur une relation de dominance (ou approches Pareto), et les approches basées sur un indicateur.

2.1.3.1.1 Approches scalaires. Cette stratégie consiste à réduire le problème d'optimisation multiobjectif original en un problème d'optimisation monoobjectif. L'exemple le plus simple et le plus populaire consiste à agréger les n fonctions objectif en une seule valeur scalaire à l'aide

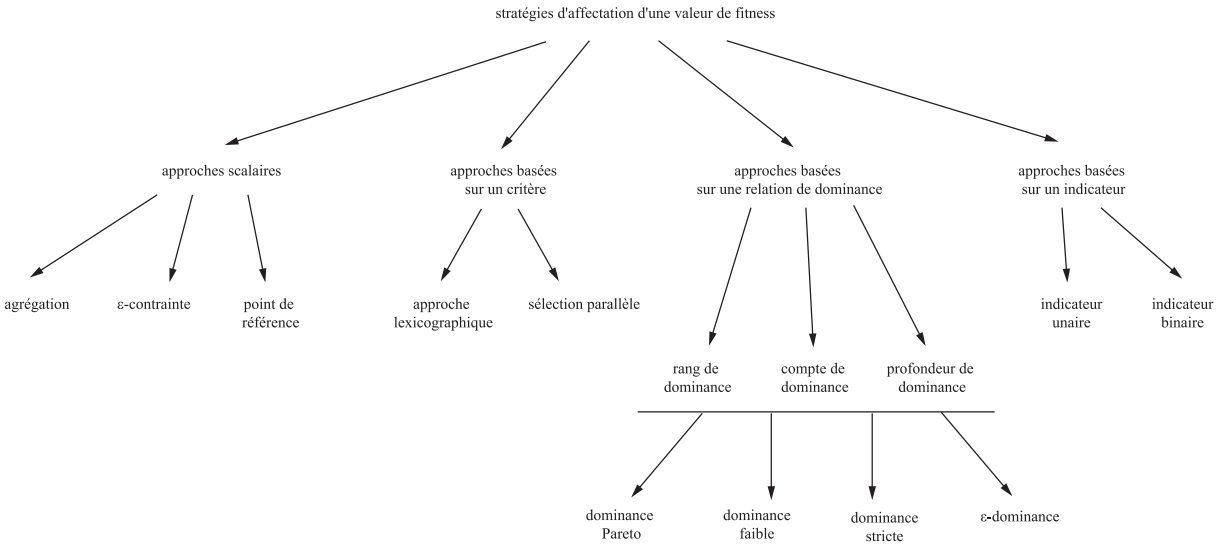


Fig. 2.1 – Classification des stratégies d'affectation d'une valeur de fitness.

d'une somme pondérée. La méthode ϵ-contrainte ou les méthodes basées sur un point de référence sont d'autres exemples d'approches scalaires (Miettinen, 1999).

2.1.3.1.2 Approches basées sur un critère. Chaque fonction objectif est ici traitée séparément. Par exemple, dans l'algorithme VEGA (*vector evaluated genetic algorithm*) proposé par Schaffer (1985), une sélection parallèle est effectuée afin de discerner les solutions de la population courante en fonction des valeurs sur une seule fonction objectif, indépendamment des autres (Fig. 2.2). Dans les méthodes lexicographiques (Fourman, 1985), un ordre hiérarchique est défini par le décideur entre les fonctions objectif. Ces fonctions sont alors traitées les unes à la suite des autres suivant cet ordre préétabli (Fig. 2.3).

2.1.3.1.3 Approches basées sur une relation de dominance. Une relation de dominance, telle que la dominance Pareto, est définie pour classer les solutions de la population. Cette idée a pour la première fois été introduite par Goldberg (1989) au sein d'algorithmes génétiques. Trois stratégies principales peuvent être distinguées (Zitzler *et al.*, 2004).

- **Rang de dominance** (*dominance-rank*). Cette technique consiste à calculer le nombre d'éléments de la population courante qui dominent une solution donnée (Fonseca et Fleming, 1993). Le rang de dominance a pour la première fois été utilisé au sein de l'algorithme MOGA (*multiobjective genetic algorithm*) de Fonseca et Fleming (1993), où la valeur de fitness d'une solution est égale à son rang de dominance au sein de la population, incrémenté de un.
- **Compte de dominance** (*dominance-count*). Ici, la valeur de fitness d'une solution correspond au nombre d'individus qui sont dominés par cette solution.
- **Profondeur de dominance** (*dominance-depth*). Cette stratégie consiste à classer les solutions de la population en différentes classes, ou fronts (Goldberg, 1989). Une solution qui appartient à une classe ne domine aucun individu de cette même classe. Ainsi, les éléments du premier front appartiennent au meilleur ensemble non-dominé ; ceux du deuxième front au deuxième meilleur ensemble non-dominé, et ainsi de suite. Cette dernière approche est utilisée dans

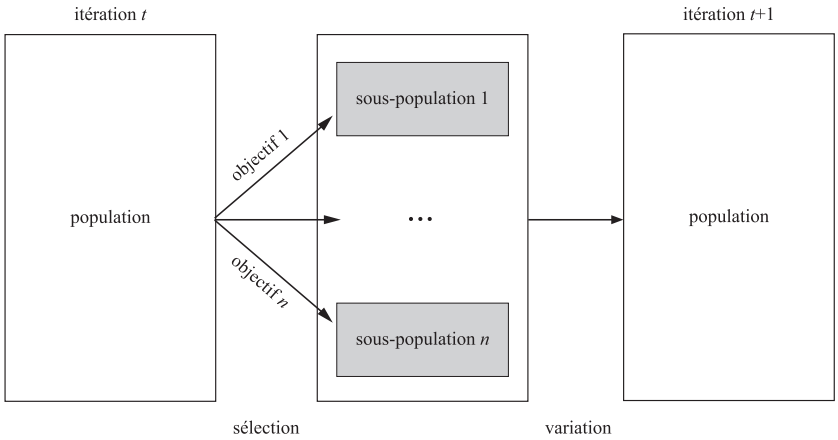


Fig. 2.2 – Sélection parallèle.

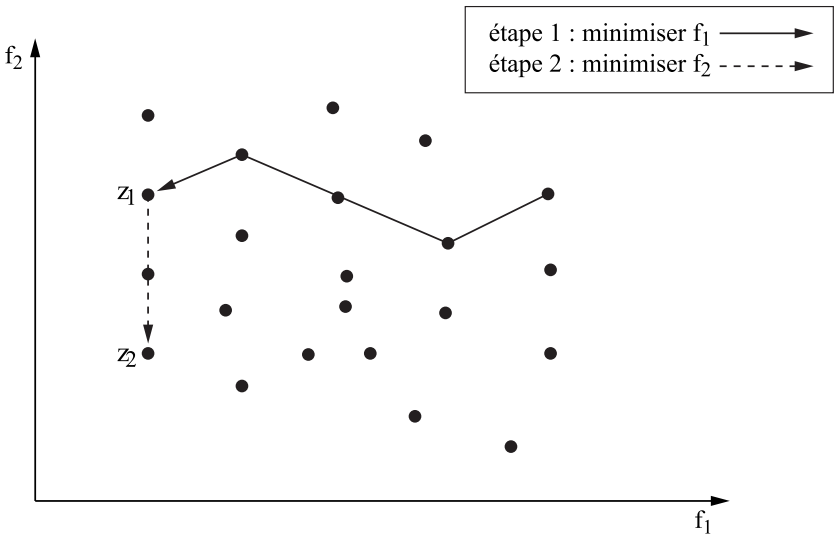


Fig. 2.3 – Ordre lexicographique pour un problème à deux objectifs.

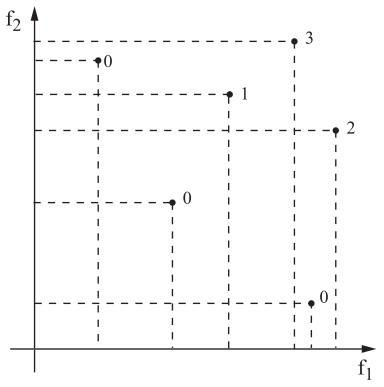


Fig. 2.4 – Rang de dominance.

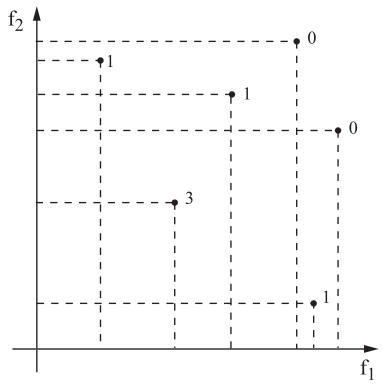


Fig. 2.5 – Compte de dominance.

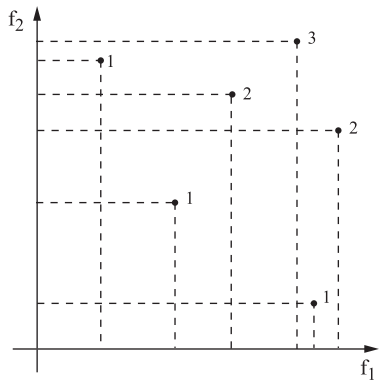


Fig. 2.6 – Profondeur de dominance.

les algorithmes NSGA (*non-dominated sorting genetic algorithm*) (Srinivas et Deb, 1994) et NSGA-II (Deb *et al.*, 2002).

Ces trois approches sont respectivement illustrées sur les figures 2.4, 2.5 et 2.6. Différentes approches peuvent également être hybridées, ce qui est le cas, par exemple, dans SPEA (Zitzler et Thiele, 1999) ou SPEA2 (Zitzler *et al.*, 2001b), où la profondeur de dominance est combinée au rang de dominance.

Dans le cadre des approches basées sur une relation de dominance, la dominance Pareto est la plus couramment utilisée. Néanmoins, n'importe quelle relation peut être utilisée, telle que les relations de dominance faible, stricte, etc, définies dans la section 1.1.3. Ainsi, des techniques récentes basent l'affectation des valeurs de fitness sur l'ϵ-dominance (Deb *et al.*, 2005a) ou sur la notion de g-dominance, comme proposé par Molina *et al.* (2009).

2.1.3.1.4 Approches basées sur un indicateur. Pour ce type d'approche, le but de l'optimisation est formulé à l'aide d'un indicateur de qualité tel que défini dans la section 1.3.1. Le choix de l'indicateur représente l'objectif général du processus de recherche. Ainsi, les valeurs de fitness sont calculées en comparant les individus sur la base de cet indicateur de qualité I. En règle générale, aucun mécanisme de préservation de la diversité n'est nécessaire avec ce type d'approche, dépendamment de l'indicateur utilisé.

Par exemple, Zitzler et Künzli (2004) proposent de formuler le but de l'optimisation à l'aide d'un indicateur binaire de qualité $I : \Omega \times \Omega \rightarrow \mathbb{R}$ comme suit :

$$\arg\min_{A \in \Omega} I(A, R), \tag{2.1}$$

où R est un ensemble de référence. Il n'est pas requis que cet ensemble de référence R soit connu, il sert juste à la formalisation du but de l'optimisation. Ainsi, R étant fixe, l'indicateur I peut être vu comme une fonction unaire qui affecte, à chaque approximation $A \in \Omega$, une I-valeur reflétant sa qualité par rapport au but de l'optimisation. Si I préserve la dominance (Zitzler et Künzli, 2004), $I(A, R)$ est minimum pour $A = R$.

D'autres approches basées sur un indicateur sont fondées sur un indicateur unaire de qualité. En particulier, les algorithmes visant à maximiser l'hypervolume de l'approximation retournée constituent un champ de recherche très actif au sein de la communauté. On peut citer l'algorithme

SMS-EMOA (*S-metric selection EMO algorithm*) (Emmerich *et al.*, 2005; Beume *et al.*, 2007), ou encore l'algorithme HypE (*hypervolume estimation algorithm for multiobjective optimization*) proposé par Bader et Zitzler (2008). Afin de diminuer le temps de calcul, ces derniers sont souvent utilisés en conjonction avec une approche basée sur une relation de dominance.

2.1.3.2 Préservation de la diversité

Comme précédemment remarqué, trouver une approximation de l'ensemble Pareto optimal n'est pas uniquement une question de convergence. L'approximation finale doit également être bien répartie dans l'espace objectif. Toutefois, les méthodes classiques d'affectation de la valeur de fitness présentées ci-dessus ont souvent tendance à converger de façon prématurée vers le front Pareto, ce qui ne garantit pas l'uniformité de l'ensemble trouvé. Afin d'éviter ce problème, un mécanisme de préservation de la diversité, basé sur une mesure de distance, doit être intégré au sein de la méthode de résolution afin de distribuer la population de manière uniforme le long du front Pareto. Dans le cadre de l'optimisation multiobjectif, cette mesure de distance est habituellement basée sur la distance euclidienne entre les vecteurs objectif. Néanmoins, cette mesure peut également être définie dans l'espace décisionnel, ou peut même combiner les deux espaces. Comme suggéré par Zitzler *et al.* (2004), les techniques de préservation de la diversité peuvent très bien être classées dans des catégories identiques à celles qui sont utilisées par les méthodes statistiques d'estimation de densité (Silverman, 1986).

2.1.3.2.1 Méthodes par noyau. Les méthodes par noyau définissent le voisinage d'une solution par rapport à une fonction noyau k prenant une mesure de distance comme argument (Fig. 2.7). En pratique, pour chaque solution, on calcule la distance qui la sépare de chacun des autres éléments de la population. La fonction noyau k est alors appliquée à chaque distance, et les valeurs obtenues sont ensuite sommées. La somme des valeurs de la fonction k représente l'estimation de densité pour l'individu correspondant. Le *fitness sharing*, initialement proposé par Goldberg et Richardson (1987), est probablement la technique par noyau la plus populaire dans le cadre des algorithmes évolutionnaires. Elle est, par exemple utilisée dans les algorithmes MOGA (Fonseca et Fleming, 1993), NPGA (Horn *et al.*, 1994) et NSGA (Srinivas et Deb, 1994).

2.1.3.2.2 Méthodes du plus proche voisin. Dans les méthodes du plus proche voisin, la distance entre une solution et son $k$ème plus proche voisin est prise en compte pour estimer la densité d'une solution (Fig. 2.8). Dans SPEA2 (Zitzler *et al.*, 2001b), l'estimateur de densité est basé sur l'inverse de cette distance. La préservation de la diversité opérée au sein de NSGA-II (Deb *et al.*, 2002) est quant à elle basée sur la distance de *crowding*, initialement proposée par Holland (1975) et utilisée par De Jong (1975) pour éviter une éventuelle *dérive génétique*. Cette mesure estime la densité d'une solution par rapport au volume de l'hyper-rectangle défini par ses plus proches voisins directs, une valeur de diversité infinie étant donc affectée aux solutions extrêmes.

2.1.3.2.3 Histogrammes. Ce dernier type d'approche consiste à partitionner l'espace de recherche par une hyper-grille (Fig. 2.9). La densité autour d'une solution est alors tout simplement estimée par le nombre d'éléments de la population avec qui elle partage la même case de la grille. Cette hyper-grille peut être statique, mais elle peut également être adaptée au cours de la

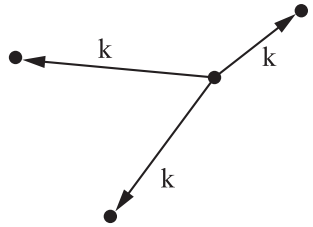


Fig. 2.7 – Méthode par noyau.

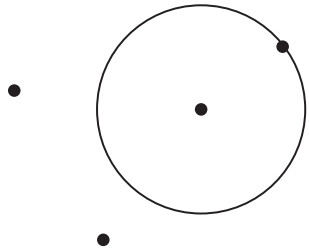


Fig. 2.8 – Méthode du plus proche voisin.

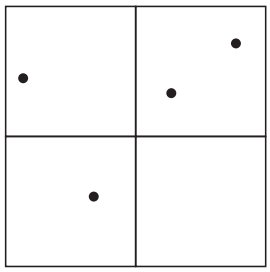


Fig. 2.9 – Histogramme.

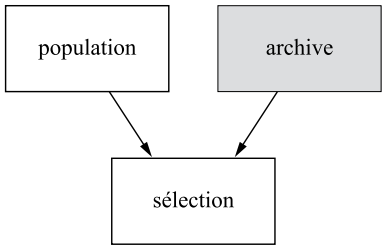


Fig. 2.10 – Sélection élitiste : l'archive participe activement au processus de recherche.

recherche par rapport à la population courante. Cette approche est utilisée dans les algorithmes PAES (Knowles et Corne, 2000) et PESA (Corne *et al.*, 2000).

2.1.3.3 Élitisme

L'élitisme a pour but principal de préserver les solutions non-dominées trouvées par l'algorithme au cours de la recherche. De la façon la plus simple, ceci peut se faire lors de l'étape de remplacement de la P-META considérée, par le biais de la population principale. D'autre part, la totalité, ou un sous-ensemble des solutions non-dominées peuvent également être sauvegardées au sein d'une population secondaire : l'*archive*. Par ailleurs, bien qu'initialement conçue pour être utilisée de façon passive (l'archive est séparée du processus de recherche), cette dernière peut également être utilisée de façon active lors d'une étape de sélection afin de générer de nouvelles solutions (Fig. 2.10). La difficulté principale réside néanmoins dans la gestion de cette archive et dans la façon dont elle est mise à jour à chaque itération de l'algorithme.

2.1.3.3.1 Gestion de l'archive. Dans un premier temps, notez que la mise à jour du contenu de l'archive à l'aide de nouvelles solutions potentiellement non-dominées trouvées par l'algorithme repose généralement sur la relation de dominance Pareto. Mais, dans la littérature, d'autres types de relation peuvent être utilisés en lieu et place de la dominance Pareto. Plusieurs exemples ont déjà été évoqués. On peut distinguer quatre stratégies d'archivage qui dépendent principalement de critères de taille, mais également de convergence et de diversité (Fig. 2.11) :
- *aucune archive* ;
- une *archive de taille non-bornée* ;
- une *archive de taille bornée* ;
- une *archive de taille fixe*.

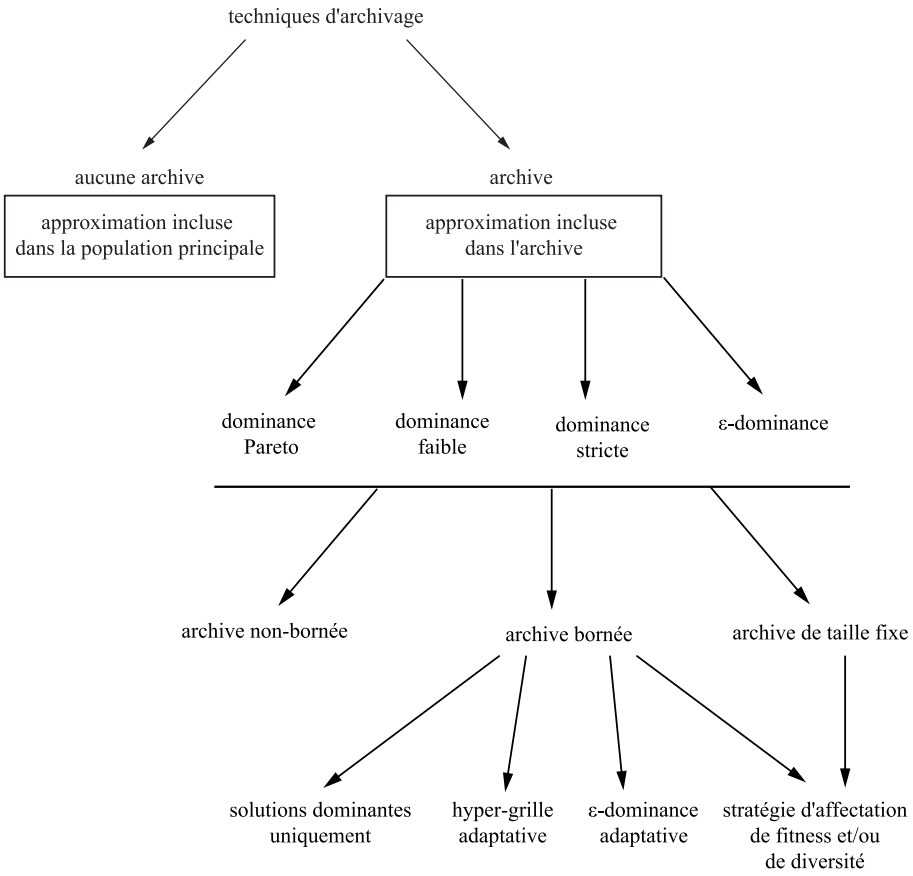


Fig. 2.11 – Classification des stratégies de gestion de l'archive.

Premièrement, si l'approximation courante est contenue au sein de la population principale, le maintient d'une archive n'est pas nécessaire. D'autre part, si une archive est maintenue, elle comprend généralement l'approximation de l'ensemble Pareto optimal, car les solutions dominées ne sont pas sauvegardées. Ainsi, une archive non-bornée peut être utilisée afin de préserver l'ensemble des solutions non-dominées trouvées au cours du processus de recherche. Cependant, certains problèmes contiennent un nombre exponentiel de solutions non-dominées (parfois infini pour des problèmes continus), il est tout simplement impossible de toutes les sauvegarder. Par conséquent, des opérations supplémentaires doivent être mises en place afin de réduire le nombre de solutions stockées. La gestion d'archives bornées est discutée par Knowles et Corne (2004).
Une stratégie courante est de borner la taille de l'archive par rapport à une technique d'affectation des valeurs de fitness, ou de préservation de la diversité. Par exemple, les solutions peuvent être classées à l'aide de la distance de *crowding*. De même, une forme relaxée de dominance, telle que l'ϵ-dominance, peut être utilisée pour maintenir l'archive. Ceci permet de réduire la taille de l'archive avec une complexité en temps de calcul comparable. Néanmoins, il s'avère souvent difficile de fixer une ϵ-valeur appropriée. Il est tout de même possible de mettre cette valeur à jour de façon adaptative (Laumanns *et al.*, 2001), mais ceci a pour effet d'augmenter la complexité algorithmique. Par ailleurs, ce concept peut s'avérer dangereux et destructeur pour des problèmes

dont le nombre de solutions réalisables est faible. Une autre stratégie simple se base sur le principe suivant (Rudolph et Agapie, 2000). Tant que l'archive est au-dessous de sa capacité, toutes les solutions non-dominées sont intégrées. Au contraire, si la taille de l'archive a atteint son plus haut niveau, seules les solutions dominant au moins une solution de l'archive sont acceptées. Dans les deux cas, toute solution dominée est écartée. Rudolph et Agapie (2000) ont également proposé de mettre l'archive à jour selon l'âge de chaque solution archivée, c'est à dire le nombre d'itérations écoulé depuis que la solution est présente dans l'archive.
Pour finir, une dernière technique d'archivage se base sur une capacité de stockage de taille fixe, où un mécanisme d'archive bornée est utilisé quand il y a trop de solutions archivées, et où des solutions dominées sont intégrées dans l'archive si l'ensemble est de trop petite taille. Ceci est fait par exemple dans SPEA2 (Zitzler *et al.*, 2001b).

2.1.4 Conception d'algorithmes évolutionnaires multiobjectif

Un algorithme évolutionnaire (Eiben et Smith, 2003) est une méthode de recherche qui appartient à la classe de métaheuristiques à base de population (Talbi, 2009). Une population de solutions est améliorée de façon itérative par le biais d'opérateurs stochastiques. À partir d'une population initiale, chaque individu est évalué dans l'espace objectif et une stratégie de sélection permet de construire une population Parent. Des opérateurs de variation (croisement, mutation) sont alors appliqués afin de créer une population Enfant. Ensuite, une stratégie de remplacement détermine quels individus survivent au sein des populations Parent et Enfant. Une telle itération représente une génération. Le processus est itéré jusqu'à la satisfaction d'un critère d'arrêt.
L'adaptation des algorithmes évolutionnaires pour l'optimisation multiobjectif porte principalement sur l'affectation d'une valeur de fitness, la préservation de la diversité et l'élitisme. Les étapes d'affectation d'une valeur de fitness et de préservation de la diversité sont nécessaires à la discrimination des individus aux cours de la sélection et de remplacement de l'algorithme évolutionnaire. La notion d'élitisme apparaît quant à elle lors de l'étape de remplacement, où les meilleures solutions peuvent être sauvegardées au sein de la population principale, ou lors d'une nouvelle étape de mise à jour de l'archive dans laquelle les meilleures solutions sont intégrées. La mise à jour du contenu de l'archive peut apparaître à chaque itération de l'algorithme évolutionnaire. Enfin, l'étape de sélection peut également être élitiste si des solutions provenant de l'archive sont choisies pour devenir Parent.

2.1.4.1 Un modèle de conception unifié

Quel que soit le problème à résoudre, les principaux composants prenant part dans la conception d'un algorithme évolutionnaire multiobjectif sont donc les suivants :

1. Concevoir une **représentation** (voir la section 2.1.2.1).
2. Concevoir une stratégie d'**initialisation de la population**.
3. Concevoir une fonction d'**évaluation** (voir la section 2.1.2.2).
4. Concevoir des **opérateurs de variation** appropriés.
5. Déterminer une stratégie d'**affectation d'une valeur de fitness** (voir la section 2.1.3.1).
6. Déterminer une stratégie de **préservation de la diversité** (voir la section 2.1.3.2).
7. Déterminer une stratégie de **sélection**.

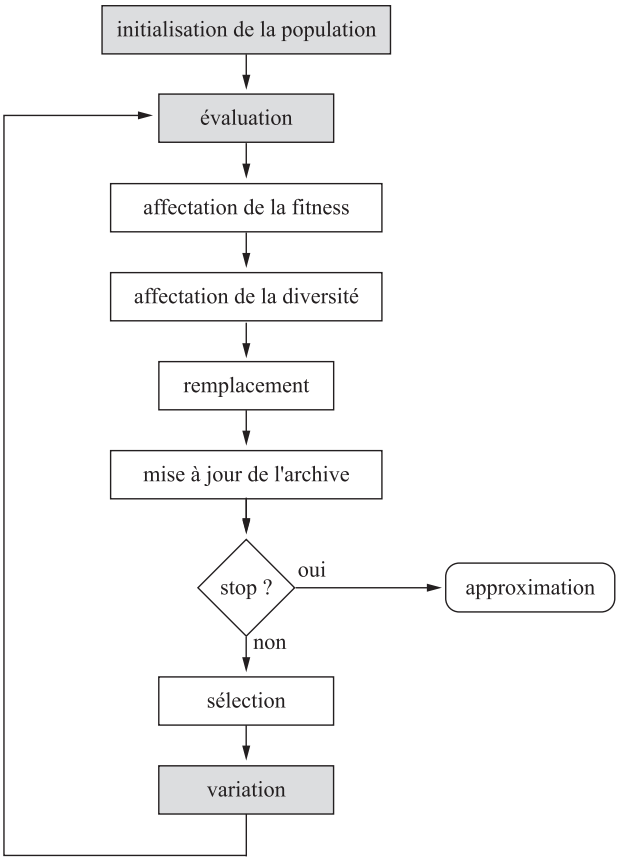


Fig. 2.12 – Étapes d'un algorithmes évolutionnaire multiobjectif.

8. Déterminer une stratégie de **remplacement**.
9. Déterminer une stratégie de **gestion de l'archive** (voir la section 2.1.3.3).
10. Déterminer une **condition d'arrêt**.

Lors de la conception d'une métaheuristique, on peut toujours distinguer les composants spécifiques au problème traité et les composants génériques. Ainsi, les quatre premiers composants présentés ci-dessus dépendent fortement du problème à résoudre, tandis que les six derniers peuvent être considérés comme indépendants (même si certaines stratégies spécifiques peuvent également être envisagées). Notez que les notions de représentation et d'évaluation sont partagées par toute métaheuristique, que les notions d'initialisation de la population et de critère d'arrêt sont partagées par toute métaheuristique à base de population, que les notions d'opérateurs de variation, de sélection et de remplacement sont partagées par tout algorithme évolutionnaire, et que les notions d'affectation d'une valeur de fitness, de préservation de la diversité et de gestion de l'archive sont partagées par toute métaheuristique pour l'optimisation multiobjectif.
En suivant cette décomposition à grain fin, un modèle de conception unifié pour les algorithmes évolutionnaire multiobjectif est proposé et présenté dans la figure 2.12. Les composants spécifiques au problème apparaissent en gris sur la figure. L'algorithme 1 illustre la trame de haut-niveau d'un algorithme évolutionnaire multiobjectif général.

Algorithme 1 Modèle d'un algorithme évolutionnaire multiobjectif

1. **Initialisation.** Démarrer avec une population initiale P de taille N fournie en paramètre, ou la générer de façon aléatoire.
2. **Évaluation.** Évaluer les solutions de la population P dont les valeurs objectif sont inconnues.
3. **Affectation d'une valeur de fitness.** Calculer les valeurs de fitness pour chaque solution x de la population P.
4. **Affectation d'une valeur de diversité.** Calculer les valeurs de diversité pour chaque solution x de la population P.
5. **Sélection pour le remplacement.** Supprimer N' solutions de la population P afin d'obtenir une population de taille N.
6. **Mise à jour de l'archive.** Mettre à jour l'archive A à l'aide des solutions de la population principale P.
7. **Condition d'arrêt.** Si une condition d'arrêt est satisfaite, retourner les solutions non-dominées de l'archive A et/ou de la population principale P. Stop.
8. **Sélection pour la reproduction.** Sélectionner N' individus parmi la population P et ajouter les solutions sélectionnées dans une population temporaire P'.
9. **Variation.** Appliquer les opérateurs de croisement et de mutation aux solutions de la population temporaire P' et ajouter les solutions résultantes à la population principale P. Aller à l'étape 2.

2.1.4.2 Description des composants

Cette section présente les composants spécifiques aux algorithmes évolutionnaires multiobjectif, à base de population, qui n'ont pas encore été présentés précédemment : initialisation, variation, sélection, remplacement, et critère d'arrêt.

2.1.4.2.1 Initialisation. Quelle que soit la solution algorithmique considérée, une manière d'initialiser une solution (ou une population de solutions) est attendue. Lorsque l'on fait face à une métaheuristique à base de population, il convient de garder à l'esprit que la population initiale doit être bien diversifiée dans l'espace de recherche, ceci afin d'éviter une convergence prématurée. Cette remarque est d'autant plus vraie pour l'optimisation multiobjectif, où le but est de trouver une approximation de bonne qualité aussi bien en termes de convergence que de diversité. La façon d'initialiser une solution est étroitement liée au problème à résoudre et à la représentation choisie. Dans la plupart des cas, la population initiale est générée de façon aléatoire.

2.1.4.2.2 Variation. Le but des opérateurs de variation est de modifier la représentation des solutions en vue de progresser dans l'espace de recherche. Dans le cadre d'algorithmes évolutionnaires, ces composants liés au problème sont des opérateurs stochastiques. De façon générale, on peut distinguer deux types d'opérateur de variation : les opérateurs de croisement et les opérateurs de mutation. Les opérateurs de croisement sont pour la plupart des opérateurs binaires

et, parfois n-aires. Leur rôle est d'hériter de caractéristiques des deux solutions Parent afin de générer une ou deux solutions Enfant. Les opérateurs de mutation sont quant à eux unaires car ils agissent sur une solution unique. Ils consistent à effectuer une légère modification de certains individus de la population.

2.1.4.2.3 Sélection. L'étape de sélection est l'un des principaux opérateurs de recherche de l'algorithme évolutionnaire. Elle consiste à choisir des solutions de la population courante qui seront utilisées pour générer la population Enfant. En général, meilleure est une solution, plus élevée sera sa chance d'être sélectionnée. Les stratégies classiques sont les tournois stochastiques ou déterministes, la roulette, la sélection aléatoire, etc. Il existe une stratégie élitiste spécifique à l'optimisation multiobjectif qui consiste à inclure des solutions provenant de l'archive lors du processus de sélection. Ainsi, ces solutions élites contribuent également à l'évolution. Une telle approche a été appliquée avec succès dans divers algorithmes élitistes tels que SPEA (Zitzler et Thiele, 1999), SPEA2 (Zitzler *et al.*, 2001b) ou encore PESA (Corne *et al.*, 2000). En outre, afin d'interdire le croisement de deux solutions Parent trop différents l'un de l'autre dans l'espace de recherche (ce qui a plus de chance d'arriver en optimisation multiobjectif), la restriction d'accouplement (Goldberg, 1989) peut également être mentionnée comme une stratégie intéressante à intégrer à la conception d'un algorithme évolutionnaire multiobjectif.

2.1.4.2.4 Remplacement. La pression de sélection est également affectée à l'étape de remplacement, où les survivants sont sélectionnés au sein de la population courante et de la population Enfant. Dans le cas d'un remplacement générationnel, la population Enfant devient systématiquement la population courante. Une stratégie de remplacement élitiste consiste à conserver les N meilleures solutions des deux populations, où N est la taille requise de la population. Pour ce faire, deux techniques existent : une réduction directe et une réduction itérative, où la moins bonne solution est à plusieurs reprises supprimée jusqu'à ce que la taille de la population nécessaire soit atteinte. Ainsi, à chaque suppression, les informations de fitness et de diversité des individus restants sont mises à jour.

2.1.4.2.5 Condition d'arrêt. Comme une méthode itérative calcule des approximations successives, un test pratique est nécessaire pour déterminer à quel moment le processus doit s'arrêter. Les exemples les plus classiques consistent à stopper l'algorithme évolutionnaire au bout d'un certain nombre d'itérations, d'un certain nombre d'évaluations, d'un certain temps d'exécution, etc. Une stratégie plus évoluée consiste à recenser le nombre d'itérations consécutives sans amélioration, et d'arrêter le processus de recherche quand il dépasse une valeur donnée en paramètre. Mais une itération non-améliorante est difficile à formuler en optimisation multiobjectif. Par exemple, une itération peut être définie comme non-améliorante si le nombre de nouvelles solutions non-dominées ajoutées à l'archive est nul.

2.1.4.3 Algorithmes existants comme instances du modèle

Par le biais du modèle de conception unifié présenté ci-dessus, nous affirmons qu'un grand nombre d'algorithmes évolutionnaires multiobjectif de référence proposés durant les deux dernières décennies sont basées sur de simples variations des composants indépendants du problème. Dans le tableau 2.1, trois méthodologies, à savoir NSGA-II (Deb *et al.*, 2002), SPEA2 (Zitzler *et al.*,

TABLE 2.1 – Algorithmes évolutionnaires multiobjectif existants comme instances du modèle unifié proposé. Trois exemples sont présentés : NSGA-II (Deb *et al.*, 2002), SPEA2 (Zitzler *et al.*, 2001b) et IBEA (Zitzler et Künzli, 2004).

composants	NSGA-II	SPEA2	IBEA
fitness	profondeur de dominance (dominance Pareto)	compte et rang de dominance (dominance Pareto)	indicateur binaire de qualité
diversité	distance de *crowding*	plus proche voisin	aucun
sélection	tournoi binaire	élitiste (tournoi binaire)	tournoi binaire
remplacement	remplacement élitiste	remplacement générationnel	remplacement élitiste
gestion de l'archive	aucun	archive de taille fixe	aucun
arrêt	nombre de générations	nombre de générations	nombre de générations

2001b) et IBEA (Zitzler et Künzli, 2004), sont considérées comme de simples instances du même modèle. Bien sûr, seuls les composants indépendants du problème sont présentés pour des raisons évidentes. NSGA-II et SPEA2 sont probablement les deux algorithmes évolutionnaires multiobjectif les plus fréquemment rencontrés dans la littérature, soit pour faire face à un problème original, soit pour servir de référence lors d'une comparaison de différentes approches. Concernant IBEA, c'est une bonne illustration de la nouvelle tendance portant sur la recherche basée sur un indicateur de qualité, qui commence à devenir populaire depuis quelques années.

Exemple 2.1 (NSGA-II) (Deb *et al.*, 2002)
NSGA-II est probablement l'algorithme évolutionnaire multiobjectif le plus populaire. À chaque génération, les solutions de la population courante sont classées en fonction de deux critères. Deux valeurs sont donc affectées à chaque membre de la population. Tout d'abord, la stratégie d'affectation des valeurs de fitness se base sur le compte de dominance et sur la dominance Pareto. Toutes les solutions non-dominées de la population se voient affecter une valeur de fitness de 1 (premier front), puis elles sont retirées de la population. Toutes les solutions non-dominées de la population se voient affecter une valeur de fitness de 2 (deuxième front), puis elles sont retirées de la population. Et ainsi de suite. Ce processus est itéré jusqu'à ce que l'ensemble des solutions dont la valeur de fitness est à évaluer soit vide. Cette première valeur représente la qualité de la solution en termes de convergence. La deuxième valeur consiste à estimer la densité de solutions autour d'un point particulier de l'espace objectif à l'aide de la distance de *crowding*, et représente la qualité de la solution en termes de diversité. La distance de *crowding* est calculée parmi les solutions appartenant à un même front (c'est-à-dire ayant une valeur de fitness identique). Ainsi, une solution est considérée comme plus performante qu'une autre si elle a une meilleure valeur de fitness, ou en cas d'égalité, si elle a la meilleure distance de *crowding*. À chaque itération de l'algorithme évolutionnaire, N solutions Enfant sont générées, où N est la taille de la population. La stratégie de sélection consiste en un tournoi déterministe réalisé entre deux solutions choisies aléatoirement. La phase de remplacement élitiste fonctionne comme suit. La population de l'itération précédente et la population Enfant sont fusionnées, et seules les meilleures solutions survivent selon la procédure hiérarchique énoncée ci-dessus. Par ailleurs, dans la version originale de NSGA-II, aucune technique d'archivage n'est considérée, l'approximation courante de l'ensemble Pareto optimal étant contenue au sein de la population principale.

Exemple 2.2 (SPEA2) (Zitzler *et al.*, 2001b)
SPEA2 est une extension de l'algorithme SPEA, où une stratégie d'affectation des valeurs de fitness améliorée est proposée. SPEA2 gère intrinsèquement une archive interne de taille fixe qui est utilisée au cours de l'étape de sélection pour créer des solutions Enfant. Lors d'une itération, à chaque membre de la population courante et de l'archive x est attribuée une valeur $S(x)$ égale au nombre de solutions qu'elle domine en termes de dominance Pareto (compte de dominance). Ensuite, la valeur de fitness $F(x)$ de la solution x est calculée en additionnant les valeurs $S(x)$ de toutes les solutions dominées par x. En outre, une stratégie de préservation de la diversité, basée sur une technique du $k$ème plus proche voisin, est incorporée. L'étape de sélection est élitiste, et se compose d'un tournoi binaire avec remplacement appliqué sur l'archive uniquement. Enfin, étant donné que l'archive de SPEA2 a une capacité de stockage de taille fixe, un mécanisme d'archive bornée, basé sur les informations de fitness et de diversité, est utilisé lorsque la taille de l'ensemble non-dominé est trop élevée. À l'inverse, lorsque la taille de l'archive est trop faible, des solutions dominées sont incorporées. Par ailleurs, les solutions constituant l'approximation courante étant contenues dans l'archive, et l'étape de sélection étant appliquée sur cette même archive uniquement, une simple stratégie de remplacement générationnel est considérée.

Exemple 2.3 (IBEA) (Zitzler et Künzli, 2004)
L'idée principale introduite par IBEA est d'instaurer un ordre total entre les solutions de la population courante en généralisant la relation de dominance au moyen d'un indicateur binaire de qualité arbitraire $I : \Omega \times \Omega \to \mathbb{R}$. Cet indicateur représente le but de l'optimisation. La véritable question est donc de savoir comment l'intégrer au sein d'une stratégie d'affectation des valeurs de fitness. En d'autres termes, la valeur de fitness d'une solution x appartenant à la population courante P doit refléter son utilité à l'égard du but de l'optimisation. Ainsi, à chaque individu est attribuée une valeur de fitness $F(x)$ mesurant la « perte de qualité » si x était retiré de la population courante. Plusieurs techniques sont discutées par Zitzler et Künzli (2004), et celle que retiennent les auteurs est une approche additive, basée sur une comparaison deux à deux des solutions de la population courante, et qui amplifie l'influence des solutions non-dominées sur les solutions dominées (2.2).

$$F(x) = \sum_{x' \in P \setminus \{x\}} -e^{-I(x',x)/\kappa} \tag{2.2}$$

où $\kappa > 0$ est un facteur d'échelle. Une description détaillée de l'algorithme IBEA est reproduite dans l'algorithme 2. La sélection pour la reproduction se compose d'un tournoi binaire déterministe. La sélection pour le remplacement consiste à supprimer, de façon itérative, la plus mauvaise solution de la population courante jusqu'à ce que la taille requise soit atteinte ; les valeurs de fitness des individus restants étant mises à chaque suppression.
Ainsi, n'importe quel indicateur binaire de qualité peut être utilisé dans (2.2). À titre d'exemple, Zitzler et Künzli (2004) définissent l'indicateur ϵ-additif ($I_{\epsilon+}$) comme suit :

$$I_{\epsilon+}(z, z') = \max_{i \in \{1,\ldots,n\}} \{z_i - z'_i\} \tag{2.5}$$

Il donne la valeur minimale par laquelle un vecteur objectif $z \in Z$ doit être, ou peut être translaté dans l'espace objectif pour faiblement dominer une autre solution $z' \in Z$ (Fig. 2.13). Notez que cette translation peut prendre une valeur négative et que les fonctions objectif sont supposées

Algorithme 2 Modèle de l'algorithme IBEA

1. **Initialisation.** Démarrer avec une population initiale P de taille N fournie en paramètre, ou la générer de façon aléatoire.
2. **Affectation d'une valeur de fitness.** Calculer les valeurs de fitness pour chaque solution x de la population P :
$$F(x) \leftarrow \sum_{x' \in P \setminus \{x\}} -e^{-I(x',x)/\kappa} \tag{2.3}$$
3. **Sélection pour le remplacement.** Répéter les trois étapes suivantes tant que la taille de la population courante P dépasse N :
   (a) Choisir une solution $x^\star \in P$ avec la plus petite valeur de fitness : $F(x^\star) \leq F(x)$ pour tout $x \in P$.
   (b) Supprimer $x^\star$ de la population P.
   (c) Mettre à jour les valeurs de fitness des solutions restantes. Pour tout $x \in P$:
$$F(x) \leftarrow F(x) + e^{-I(x^\star,x)/\kappa} \tag{2.4}$$
4. **Condition d'arrêt.** Si une condition d'arrêt est satisfaite, retourner les solutions non-dominées de P. Stop.
5. **Sélection pour la reproduction.** Effectuer un tournoi déterministe binaire de sélection avec remplacement sur P et ajouter les solutions sélectionnées dans une population temporaire P'.
6. **Variation.** Appliquer les opérateurs de croisement et de mutation aux solutions de la population temporaire P' et ajouter les solutions résultantes à la population temporaire P. Aller à l'étape 2.

être normalisées. Un autre exemple donné par Zitzler et Künzli (2004) est l'indicateur I_{HD}, basé sur la notion d'hypervolume (Zitzler et Thiele, 1999) :

$$I_{HD}(z,z') = \begin{cases} I_H(z') - I_H(z) & \text{si } z' \succ z \\ I_H(z+z') - I_H(z) & \text{sinon} \end{cases} \tag{2.6}$$

$I_H(z)$ représente l'hypervolume de l'espace objectif dominé par un vecteur objectif $z \in Z$. $I_H(z,z')$ mesure l'hypervolume de l'espace objectif dominé par un vecteur objectif $z' \in Z$ mais pas par $z \in Z$ (Fig. 2.14). Afin de pouvoir être utilisés au sein de l'algorithme IBEA, dans (2.2), ces indicateurs doivent être rapportés au niveau de l'espace décisionnel.

$$I(x,x') = I(f(x), f(x')) \tag{2.7}$$

D'autres indicateurs de qualité pourraient facilement être intégrés au sein de l'algorithme IBEA, tels que ceux décrits par Zitzler *et al.* (2003).

Les trois exemples d'algorithmes présentés ci-dessus s'intègrent parfaitement dans notre modèle, ce qui valide la haute flexibilité de l'approche proposée. Mais d'autres exemples peuvent

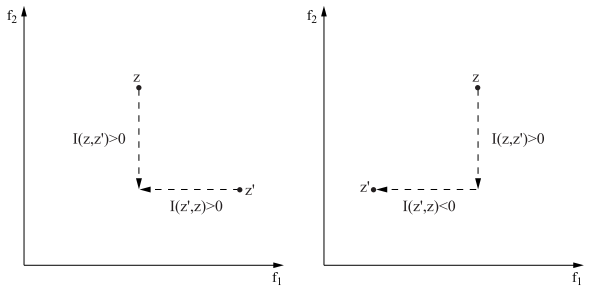


Fig. 2.13 – Illustration de l'indicateur $I_{\epsilon+}$ appliqué à deux vecteurs objectif z et $z' \in Z$. À gauche, aucune relation de dominance n'existe entre z et z' ; à droite, $z' \succ z$.

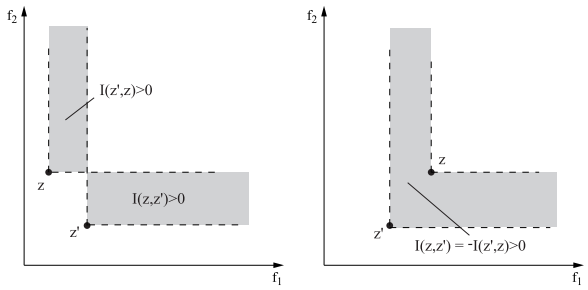


Fig. 2.14 – Illustration de l'indicateur I_{HD} appliqué à deux vecteurs objectif z et $z' \in Z$. À gauche, aucune relation de dominance n'existe entre z et z' ; à droite, $z' \succ z$.

facilement être trouvés dans la littérature. Par exemple, le seul composant qui diffère entre NSGA (Srinivas et Deb, 1994) et NSGA-II est la stratégie de préservation de la diversité, qui est basée sur le *sharing* dans NSGA et sur le *crowding* dans NSGA-II. Un autre exemple est l'algorithme ϵ-MOEA proposé par Deb *et al.* (2005a). Celui-ci est une version modifiée de NSGA-II où la relation de dominance de Pareto, utilisée lors de l'affectation des valeurs de fitness, est remplacée par une relation d'ϵ-dominance. De même, la relation de g-dominance proposée par Molina *et al.* (2009) est expérimentée par les auteurs sur une technique proche de NSGA-II, mais où la relation de dominance a été modifiée afin de prendre en compte les préférences du décideur par le biais d'un point de référence.

2.1.4.4 Un nouvel algorithme évolutionnaire pour l'optimisation multiobjectif

Dans un dernier temps, nous allons illustrer comment une telle classification générale permet également de concevoir de nouvelles méthodologies en combinant différentes idées d'approches existantes. Ainsi, ce modèle de conception unifié nous a permis de proposer un nouvel algorithme évolutionnaire multiobjectif que nous allons étudier lors de la phase expérimentale du présent chapitre. Cet algorithme, que nous avons appelé SEEA (*simple elitist evolutionary algorithm*), est basé sur une approche élitiste très simple présentée ci-dessous.

Exemple 2.4 (SEEA) (Liefooghe *et al.*, 2010)
Si l'évaluation d'une solution dans l'espace objectif ne consomme pas trop de temps de calcul (ce qui est le cas pour les problèmes traités ici), le calcul des valeurs de fitness et de diversité correspond généralement aux étapes les plus coûteuses d'un algorithme évolutionnaire multiobjectif. Partant de ce constat, nous proposons ici une méthode de recherche simple, pour laquelle aucune de ces phases n'est nécessaire, illustrée par l'algorithme 3. Ainsi, une archive de solutions potentiellement Pareto optimales est maintenue et mise à jour à chaque itération de l'algorithme. L'étape de sélection consiste en une stratégie élitiste, où les individus Parent sont sélectionnés dans l'archive uniquement. La population principale est donc créée à l'aide de l'application des opérateurs de variation sur des membres de l'archive choisis aléatoirement. Ainsi, tel que proposé par exemple par Zitzler et Thiele (1999), l'archive n'est pas seulement utilisée comme stockage externe, mais est également intégrée de façon active dans le processus de recherche au cours de la phase de sélection de l'algorithme évolutionnaire. SEEA est donc un algorithme élitiste, et

Algorithme 3 Modèle de l'algorithme SEEA

1. **Initialisation.** Démarrer avec une population initiale P de taille N fournie en paramètre, ou la générer de façon aléatoire ; initialiser l'archive A avec les solutions non-dominées de la population P.
2. **Évaluation.** Évaluer les solutions de la population P.
3. **Mise à jour de l'archive.** $A \leftarrow$ solutions non-dominées de $A \cup P$; $P \leftarrow \emptyset$.
4. **Condition d'arrêt.** Si une condition d'arrêt est satisfaite, retourner les solutions non-dominées de l'archive A. Stop.
5. **Sélection pour la reproduction.** Répéter jusqu'à ce que $|P| = N$: sélectionner aléatoirement une solution de l'archive A et ajouter la à la population P.
6. **Variation.** Appliquer les opérateurs de croisement et de mutation aux solutions de la population P. Aller à l'étape 2.

est en quelque sorte lié à d'autres algorithmes évolutionnaires élitistes tels que SPEA (Zitzler et Thiele, 1999), PESA (Corne *et al.*, 2000) ou encore SEAMO (Valenzuela, 2002). Néanmoins, contrairement à ces autres approches, aucune stratégie permettant de préserver la diversité ou de gérer la taille de l'archive n'est incluse ici, la taille de l'archive n'étant pas bornée. Notez que pour être appliqué à des problèmes d'optimisation où un nombre exponentiel, voir infini, de solutions non-dominées existent, un mécanisme additionnel devrait être conçu afin de limiter la taille de l'archive (Knowles et Corne, 2004). Un des avantages de cet algorithme est que la population (ou la taille de la population si les solutions initiales sont aléatoires) est le seul paramètre indépendant du problème. Si les solutions non-dominées sont relativement proches les unes des autres dans l'espace de décision, et si la taille de l'archive n'est pas trop petite par rapport à celle de la population principale, nous pensons que SEEA peut converger vers une bonne approximation de l'ensemble Pareto optimal en un temps d'exécution très court.

2.1.5 Conception d'algorithmes de recherche locale multiobjectif

Nous allons maintenant nous intéresser à un autre type de métaheuristique : les méthodes de recherche locale, et ceci toujours dans un contexte multiobjectif. Comparativement au nombre considérable d'algorithmes évolutionnaires multiobjectif, la quantité d'algorithmes de recherche locale multiobjectif est très restreinte, en particulier les approches basées sur une relation de dominance comme la dominance Pareto auxquelles nous allons nous intéresser ici.
Un algorithme classique de recherche locale consiste à améliorer de façon itérative une solution arbitraire par rapport à son voisinage jusqu'à ce qu'un optimum local soit atteint. Il s'agit donc d'une métaheuristique à base de solution unique (Talbi, 2009). Ainsi, la considération de ce type de méthode requiert la définition d'une structure de voisinage pour le problème traité.

Définition 2.1 (Structure de voisinage) Une *structure de voisinage* est une fonction $\mathcal{N} : X \rightarrow 2^X$ qui affecte un ensemble de solutions $\mathcal{N}(x) \subset X$ à toute solution $x \in X$. $\mathcal{N}(x)$ est le *voisinage* de x, et une solution $x' \in \mathcal{N}(x)$ est un *voisin* de x.

Étant donné que, dans le cadre de l'optimisation multiobjectif, des solutions Pareto optimales doivent être trouvées, la notion d'optimum local doit être définie en termes d'optimalité Pareto.

Définition 2.2 (Solution localement Pareto optimale) Une solution $x \in X$ est une *solution localement Pareto optimale* par rapport à une structure de voisinage $\mathcal{N}$ ssi il n'existe pas de solution voisine $x' \in \mathcal{N}(x)$ telle que $x' \succ x$.

Dans le domaine de l'optimisation multiobjectif, un algorithme de recherche de voisinage peut être utilisé comme approche autonome pour trouver une approximation de l'ensemble Pareto optimal, ou bien peut être hybridé avec d'autres métaheuristiques à base de population (voir le chapitre suivant).
Les approches de recherche locale initialement proposées pour l'optimisation multiobjectif étaient basées sur l'amélioration successive et indépendante d'une solution unique. De manière générale, l'approximation finale est contenue dans une archive externe où les solutions potentiellement Pareto optimales sont stockées. Cependant, certaines techniques récentes manipulent intrinsèquement une population de solutions évoluant en parallèle. Ces méthodes peuvent donc capitaliser les connaissances disponibles au sein de la population afin de guider et focaliser la recherche. Dans la pratique, conjuguer les principes de recherche locale avec l'utilisation d'une population semble être particulièrement prometteur pour résoudre des problèmes multiobjectif. Le terme *recherche locale à base de population* sera employé ici pour désigner ce type de méthode. Ainsi, les méthodes de recherche locales existantes qui visent à trouver une approximation de l'ensemble Pareto optimale d'un problème d'optimisation multiobjectif appartiennent à deux catégories différentes : les *approches scalaires* et les *approches basées sur une relation de dominance.*

Les approches appartenant à la première classe consistent à résoudre de multiples problèmes d'optimisation monoobjectif, en réduisant le problème orignal par le biais d'une scalarisation du vecteur de fonction objectif. La plupart des méthodes de scalarisation se basent sur une agrégation des fonctions objectif à l'aide d'une somme pondérée.

$$f_\lambda(x) = \sum_{i=1}^{n} \lambda_i f_i(x) \tag{2.8}$$

où $\lambda_i > 0$ pour tout $i \in \{1, \ldots, n\}$. Toutefois, dans le cas combinatoire, notez qu'un certain nombre de solutions Pareto optimales, connues comme *solutions non-supportées*, ne sont optimales pour aucune formulation du problème donné en (2.8). Néanmoins, d'autres techniques de scalarisation peuvent également être employées ; voir la section 2.1.3.1. Ainsi, une fois le mécanisme d'agrégation défini, prenons l'exemple d'une simple somme pondérée f_λ, le problème multiobjectif à résoudre est réduit en un problème monoobjectif et une métaheuristique classique à base de solution unique peut être appliquée jusqu'à ce qu'un optimum local, par rapport à f_λ, soit atteint. Ce processus est réitéré en modifiant le vecteur de poids λ. Pour chaque vecteur de poids, la meilleure solution trouvée est intégrée dans une archive, de laquelle les solutions dominées sont ultérieurement écartées. À cette fin, les métaheuristiques à base de solution unique vont de la recherche locale simple (Paquete et Stützle, 2003) à des méthodes plus évoluées comme la recherche tabou (Gandibleux *et al.*, 1997; Hansen, 1997) ou le recuit simulé (Serafini, 1992; Ulungu *et al.*, 1999). Quelques exemples de méthodes scalaires existantes sont MOTS (Gandibleux *et al.*, 1997; Hansen, 1997), MOSA (Ulungu *et al.*, 1999) ou TPLS (Paquete et Stützle, 2003).
La seconde classe, qui est de notre intérêt dans le présent document, consiste à définir le critère d'acceptation de la recherche locale sur une relation de dominance arbitraire, telle que la dominance Pareto. Cette classe sera notée par DMLS (*Dominance-based Multiobjective Local Search*).

L'idée de l'algorithme DMLS le plus élémentaire est de maintenir une archive de solution non-dominées, d'explorer le voisinage des membres de l'archive, et d'actualiser le contenu de la même archive à l'aide des solutions voisines explorées ; ceci jusqu'à ce que l'archive ne s'améliore plus. Dès lors, contrairement aux approches scalaires, l'archive n'est pas seulement utilisée comme stockage externe, mais correspond aussi à la population à améliorer. Ainsi, une des particularités des méthodes DMLS est qu'elles manipulent une population de taille variable.
Différentes variantes de cette idée de base ont été proposées dans la littérature. Tout d'abord, la méthode PLS-1 (*Pareto local search*) proposée par Paquete *et al.* (2004) consiste à démarrer d'une solution initiale à ajouter à l'archive. Ensuite, une solution unique est choisie au hasard parmi les membres de l'archive. La totalité des voisins de cette solution est ensuite évaluée et les solutions non-dominées sont proposées comme solutions candidates pour intégrer l'archive. Ces étapes sont itérées jusqu'à ce que plus aucune amélioration ne soit possible. Des approches très similaires ont été proposées par divers auteurs (Talbi *et al.*, 2001; Basseur *et al.*, 2003; Angel *et al.*, 2004). La seule différence est que le voisinage de l'ensemble des membres de l'archive est ici exploré, et non celui d'une solution unique. En conséquence, une itération est généralement beaucoup plus coûteuse en temps. Ensuite, l'algorithme PAES (Pareto archived evolution strategy), proposé par Knowles et Corne (2000), est une stratégie d'évolution de recherche locale où un voisin aléatoire unique est évalué à partir d'une solution unique provenant de l'archive. Des variantes de PAES où plus d'une solution archivée est sélectionnée, et où plus d'un voisin est généré par solution ont également été étudiées. Une autre différence notable est que PAES utilise une archive de taille bornée. Enfin, différents algorithmes de recherche locale multiobjectif ont été étudiés par Aguirre et Tanaka (2005) pour des solutions représentées à l'aide de vecteurs de bits. L'un d'eux, moRBC$(|A| : 1+1)^A$ (*random bit climber using the archive for restarts*), explore le voisinage d'une solution unique provenant de l'archive jusqu'à ce qu'un voisin dominant soit trouvé, ou lorsque l'ensemble du voisinage de la solution a été examiné. Si un voisin améliorant est trouvé, celui-ci remplace la solution initiale dans une archive bornée. Ce processus est répété sur tous les membres de l'archive, jusqu'à ce qu'aucun progrès ne puisse plus être réalisé.

Dans la section suivante, nous montrons que les algorithmes de type DMLS partagent un grand nombre de composants communs. Un modèle d'unification général de cette catégorie de métaheuristiques pour l'optimisation multiobjectif est donné, et certaines questions connexes sont abordées dans le détail.
Un troisième type de méthode de recherche locale pour l'optimisation multiobjectif, ayant fait son apparition récemment, n'entre ni dans la classe des méthodes scalaires, ni dans la classe des méthodes DMLS. Il s'agit de méthodes basées sur un indicateur de qualité, et utilisant une population principale de taille fixe au cours de la recherche, et non une solution unique ou une archive, comme c'est respectivement le cas avec les algorithmes scalaires et les algorithmes DMLS. Quelques mots sont donnés à propos de ces stratégies à la suite de la présentation du modèle DMLS.

2.1.5.1 Un modèle de conception unifié

Les principaux composants de recherche prenant part à la conception d'algorithmes DMLS peuvent être définis comme suit :

1. Concevoir une **représentation** (voir la section 2.1.2.1).
2. Concevoir une stratégie d'**initialisation**.

3. Concevoir une fonction d'**évaluation** (voir la section 2.1.2.2).
4. Concevoir une **structure de voisinage** appropriée.
5. Concevoir une fonction d'**évaluation incrémentale** (si possible).
6. Déterminer une stratégie de **sélection de l'ensemble courant**.
7. Déterminer une stratégie d'**exploration du voisinage**.
8. Déterminer une stratégie de **gestion de l'archive** (voir la section 2.1.3.3).
9. Déterminer une **condition d'arrêt**.

Parmi les composants présentés ci-dessus, les cinq premiers dépendent fortement du problème à résoudre, alors que les cinq derniers peuvent être considérés comme des composants génériques. Par rapport à l'optimisation monoobjectif, les étapes d'évaluation et d'évaluation incrémentale peuvent être vues comme spécifiques à l'optimisation multiobjectif, car plusieurs fonctions doivent maintenant être calculées. Concernant les composants qui ne sont pas liés au problème, ils sont presque tous spécifiques au cas multiobjectif, à l'exception de quelques stratégies très simples. Trois structures de données s'ajoutent à ces composants afin de stocker le contenu de l'archive, l'ensemble courant, et l'ensemble candidat constitué de solutions susceptibles d'intégrer l'archive à l'étape d'archivage. Dans certains cas, si le voisinage d'une solution est évalué de façon exhaustive, celle-ci peut être marquée comme « visitée » afin d'éviter une réévaluation inutile de son voisinage. Ainsi, l'archive peut contenir à la fois des solutions visitées et non-visitées.
En suivant cette décomposition, un modèle conceptuel est présenté dans la figure 2.15 et dans l'algorithme 4. Puisque les composants liés au problème sont considérés comme existants pour le problème en question, ils n'apparaissent pas sur la figure. La méthodologie commence avec un ensemble de solutions ne se dominant pas les unes des autres fourni en entrée. Ces solutions sont utilisées pour initialiser l'archive. Ensuite, trois étapes sont itérées jusqu'à ce qu'une condition d'arrêt soit satisfaite. Tout d'abord, un sous-ensemble de solutions de l'archive est choisi pour construire l'*ensemble courant*. Ensuite, le voisinage de cet ensemble est examiné afin de construire l'*ensemble candidat*. Enfin, l'archive est mise à jour avec les solutions de l'ensemble candidat. À chaque étape, une stratégie spécifique doit être décidée afin de construire une instance du modèle DMLS.

2.1.5.2 Description des composants

Les composants liés à l'initialisation, le voisinage, l'évaluation incrémentale, la sélection de l'ensemble courant, l'exploration du voisinage et la condition d'arrêt sont présentés ci-dessous. En outre, des stratégies classiques et originales sont présentées et classées pour les étapes de sélection de l'ensemble courant et d'exploration du voisinage.

2.1.5.2.1 Initialisation. Bien que les remarques concernant l'initialisation de la population d'un algorithme évolutionnaire données en section 2.1.4.2.1 s'appliquent tout à fait aux algorithmes DMLS, une spécificité supplémentaire fait que si une solution initiale est dominée par au moins une autre solution initiale, elle sera immédiatement oubliée dès le début de l'algorithme. Pourtant, à notre connaissance, la totalité des méthodes existantes se contentent d'initialiser la population avec une seule solution, à l'instar de ce qui est fait dans le cas monoobjectif avec les méthodes de recherche locale. Cette solution initiale est généralement générée aléatoirement.

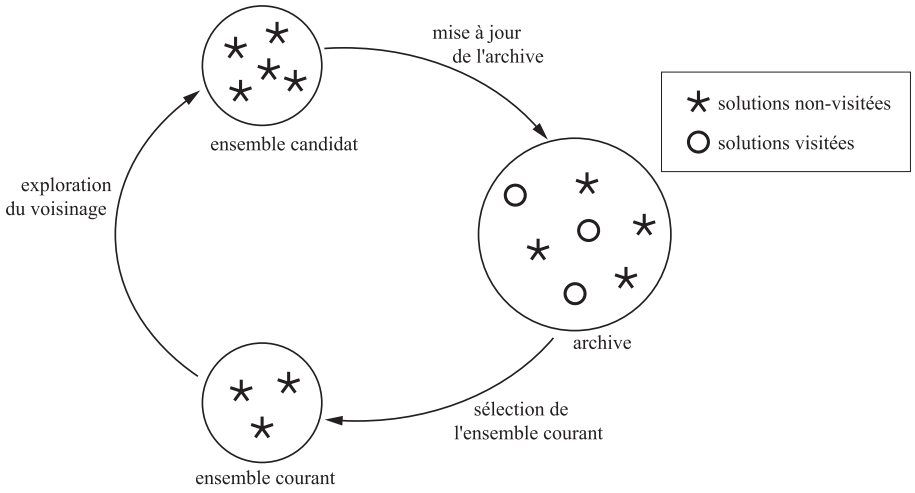


Fig. 2.15 – Étapes du modèle DMLS.

Algorithme 4 Modèle d'un algorithme DMLS

1. **Initialisation.** Démarrer avec un ensemble de solutions initiales P fourni en paramètre, ou le générer de façon aléatoire ; initialiser l'archive A avec les solutions non-dominées de P.
2. **Sélection de l'ensemble courant.** Sélectionner un ensemble de solutions non-visitées de l'archive et ajoutez-les à l'ensemble courant $P_{courant}$.
3. **Exploration du voisinage.** Pour chaque solution x de l'ensemble courant $P_{courant}$, explorer le voisinage de x selon une certaine stratégie et ajouter les solutions non-dominées explorées à l'ensemble candidat $P_{candidat}$. Dans le cas d'une exploration exhaustive du voisinage, marquer x comme visitée.
4. **Mise à jour de l'archive.** Mettre à jour l'archive A à l'aide des solutions de l'ensemble candidat $P_{candidat}$.
5. **Condition d'arrêt.** Si les solutions de l'archive A sont toutes marquées comme visitées ou si une autre condition d'arrêt est satisfaite, retourner les solutions non-dominées de l'archive A. Stop. Sinon, aller à l'étape 2.

Cependant, certaines approches consistent à construire la solution initiale à l'aide d'une heuristique spécifique, d'une méthode de construction, voire même d'une méthode exacte développée pour une variante monoobjectif du problème. Par exemple, Paquete et Stützle (2003) proposent de débuter la recherche avec une solution de très bonne qualité par rapport à l'une des fonctions objectif.

2.1.5.2.2 Voisinage. La conception d'une méthode de voisinage nécessite la définition d'une structure de *voisinage* pour le problème considéré. La définition du voisinage est une question clé de l'efficacité de la recherche locale. Une fonction de voisinage $\mathcal{N}$ affecte un ensemble de solutions $\mathcal{N}(x) \subset X$ à toute solution $x \in X$. Une solution $x' \in \mathcal{N}(x)$ est appelée un *voisin* de x. Un voisin est le résultat de l'application d'un opérateur de *mouvement* m exerçant une légère perturbation sur la solution x. Notez par ailleurs que différentes variantes d'une même recherche locale peuvent être distinguées relativement à l'ordre dans lequel les solutions voisines sont générées. En effet, cet ordre peut être déterministe, aléatoire, ou peut également varier au cours de la recherche.

2.1.5.2.3 Évaluation incrémentale. Pour des problèmes réels, l'évaluation d'une solution est souvent l'étape la plus coûteuse d'un algorithme de recherche locale (et d'une métaheuristique en général). Un moyen efficace d'évaluer une solution voisine est donc plus qu'apprécié. Cette évaluation incrémentale $\delta(x, m)$ va généralement dépendre de la solution courante x et du mouvement m à appliquer en vue de parvenir à la solution à évaluer. Cette question constitue une étape importante en termes d'efficacité, et doit donc être prise en compte lors de la conception d'un algorithme de recherche locale.

2.1.5.2.4 Sélection de l'ensemble courant. La phase initiale d'une itération d'un algorithme DMLS porte sur la sélection d'un ensemble de solutions de l'archive à partir desquelles le voisinage est à explorer. En premier lieu, notons que, si certains membres de l'archive sont marqués comme visités, ils doivent être éliminés de l'ensemble sélection pour des raisons d'efficacité évidentes. De manière générale, dans le cadre du modèle présenté ici, deux catégories principales peuvent être identifiées :
– une *sélection exhaustive*, où l'ensemble des solutions non-visitées de l'archive est sélectionné ;
– une *sélection partielle*, où uniquement un sous-ensemble de solutions est sélectionné.
Dans le premier cas, toutes les solutions vont permettre de produire un certain nombre de solutions candidates par le biais de l'opérateur de voisinage (Talbi *et al.*, 2001; Basseur *et al.*, 2003; Angel *et al.*, 2004). Dans le dernier cas, les approches existantes se contentent généralement de sélectionner un certain nombre de solutions (généralement une seule) au hasard (Paquete *et al.*, 2004). À notre connaissance, la plupart des techniques de la littérature s'inscrivent dans ces deux approches simples, mais des stratégies plus sophistiquées peuvent être conçues. Ainsi, d'une part, un sous-ensemble de solutions bien diversifié peut être sélectionné en fonction d'une mesure de densité (Sect. 2.1.3.2). C'est par exemple la solution la plus diversifiée de l'archive, par rapport à une distance de *crowding* mesurée dans l'espace objectif, qui est sélectionnée dans l'approche proposée par Aguirre et Tanaka (2005). En effet, en supposant que les solutions proches les unes des autres dans l'espace décisionnel le sont également dans l'espace objectif, nous pouvons raisonnablement affirmer qu'explorer le voisinage de solutions provenant d'une région peu fréquentée va contribuer à produire de nouvelles solutions intéressantes. D'autre part, il pourrait

également être intéressant de favoriser le choix des solutions qui ont été intégrées à l'archive lors de l'itération précédente (Knowles et Corne, 2000) ; ou au contraire, de visiter les solutions en respectant l'ordre dans lequel elles sont apparues dans l'archive.

2.1.5.2.5 Exploration du voisinage. À partir de l'ensemble courant, un certain nombre de solutions candidates doivent être générées par le biais d'une structure de voisinage. Cet ensemble candidat est obtenu par une transformation locale répétée de chaque solution contenue dans l'ensemble courant. Comme pour les stratégies de sélection de l'ensemble courant, deux catégories principales peuvent être clairement distinguées pour une solution à explorer :
– une *exploration exhaustive du voisinage*, où le voisinage d'une solution est évaluée de façon complète et déterministe ;
– une *exploration partielle du voisinage*, où seul un sous-ensemble de mouvements est appliqué.
Ces deux classes sont discutées ci-dessous.

Exploration exhaustive du voisinage. Dans ce premier cas, l'examen complet du voisinage d'une solution x est effectué. Tous les mouvements possibles sont appliqués et les solutions voisines non-dominées par rapport à x sont toutes ajoutées à l'ensemble candidat. Notez que, lors d'une exploration exhaustive du voisinage, les solutions de l'ensemble courant peuvent être marquées comme visitées à la fin de l'étape correspondante. Cela donne un avantage à cette stratégie puisque la réévaluation de certains voisins est évitée, et donc qu'un critère d'arrêt naturel peut être rencontré, voir la section 2.1.5.2.6. Cependant, cette stratégie peut paraître très coûteuse en temps et en espace de calcul en pratique, surtout si la taille du voisinage ou de l'ensemble courant est importante. Elle pourrait même s'avérer impraticable pour certaines applications où la taille du voisinage est indénombrable.

Exploration partielle du voisinage. Au sein des techniques DMLS existantes, le nombre de mouvement à appliquer lors d'une exploration partielle du voisinage est généralement défini *a priori* par une valeur fournie en paramètre. Par exemple, dans PAES (Knowles et Corne, 2000), cette valeur est généralement fixée à 1. Ainsi, si l'on considère que l'ordre des mouvements associés au voisinage d'une solution est aléatoire, l'application d'un mouvement est alors étroitement liée à la notion d'opérateur de mutation d'un point de vue évolutionnaire (Sect. 2.1.4.2.2). On voit donc bien que, dans certains cas particuliers, la différence est parfois mince entre un algorithme évolutionnaire et une P-META de recherche locale.
Cependant, des techniques plus complexes, où un tel choix serait motivé par l'état actuel de l'exploration, peuvent tout aussi bien être imaginées. En effet, l'exploration du voisinage d'une solution x pourrait continuer jusqu'à ce qu'un mouvement produise une solution voisine de meilleure qualité, ou au moins aussi bonne que x relativement à une relation de dominance donnée. Curieusement, à notre connaissance, ces techniques, bien que largement employées dans le cadre de l'optimisation monoobjectif [2], n'ont pas encore été étudiées dans une méthode DMLS, à l'exception de celle proposée par Aguirre et Tanaka (2005). Il est vrai que dans le cas monoobjectif, ce type de technique est plus facile à mettre en place du fait de l'ordre total qui existe entre les solutions. Des idées similaires ont également été proposées par Ulungu *et al.* (1999) pour un algorithme de recherche locale multiobjectif basé sur la résolution répétée d'une seule solution.

2. Ce type de stratégie est généralement dénoté comme celui du premier voisin améliorant (*first improving neighbor*) dans la littérature pourtant sur la recherche locale pour l'optimisation monoobjectif (Talbi, 2009).

Les stratégies d'exploration partielle du voisinage peuvent être divisées comme suit.
- **Voisin aléatoire**. Un seul voisin aléatoire de la solution actuelle est proposé comme candidat à l'intégration dans l'archive (Knowles et Corne, 2000).
- **Premier voisin non-dominé**. Pour chaque solution x de l'ensemble courant, le voisinage est évalué jusqu'à ce que le premier voisin non-dominé, à l'égard de x, soit trouvé. Ce voisin est alors proposé comme candidat à l'intégration de l'archive. Si aucun voisin non-dominé n'est trouvé, le processus s'arrête une fois que le voisinage de x est entièrement examiné. Dans le meilleur des cas, un seul voisin doit être évalué, alors que dans le pire des cas (tous les voisins de x sont dominés par x), le voisinage est examiné complètement, de sorte que la solution x peut être marquée comme visitée.
- **Premier voisin dominant**. Pour chaque solution courante x, les mouvements sont appliqués jusqu'à ce que l'on trouve une solution voisine dominant x (Aguirre et Tanaka, 2005). Ensuite, tous les voisins non-dominés évalués sont proposés comme solutions candidates à l'intégration au sein de l'archive. De la même manière que pour la stratégie précédente, le nombre possible de voisins à évaluer par solution de l'ensemble courant s'étend de 1 à $|\mathcal{N}(x)|$.

Bien sûr, ces trois stratégies peuvent facilement être étendues au cas où plus qu'un élément voisin aléatoire, non-dominé ou dominant doit être trouvé. En outre, une autre relation de dominance que la dominance Pareto peut être employée pour comparer les solutions deux-à-deux.
Pour les stratégies non-aléatoires, dans le pire des cas, une évaluation complète du voisinage est effectuée pour chaque solution de l'ensemble courant. Dans un tel cas, tous les mouvements sont appliqués, de sorte que la solution x correspondante peut être marquée comme visitée. Ceci a bien évidement plus de chance d'arriver pour la stratégie du premier voisin dominant. Ainsi, en supposant que les solutions dominées sont toujours écartées de l'archive, soit la solution x sera marquée comme visitée, soit elle ne sera plus incluse dans l'archive, car dominée par un de ses voisins évalués. En conséquence, cette stratégie du premier voisin dominant est la seule exploration partielle du voisinage qui donne lieu à une condition d'arrêt naturelle, ce qui évite la réévaluation inutile de solutions voisines. Pour les autres stratégies d'exploration partielle, même si des mécanismes complexes pourraient être imaginés afin de récupérer les voisins déjà évalués d'une solution donnée, ceux-ci s'avéreraient probablement inefficaces en pratique pour des voisinages de grande taille.

2.1.5.2.6 Condition d'arrêt. Outre les exemples de condition d'arrêt déjà exposés pour les algorithmes évolutionnaires, les algorithmes DMLS peuvent parfois naturellement contenir une condition d'arrêt additionnelle. En effet, lorsqu'il est possible de marquer les membres de l'archive comme visités, dépendamment de la stratégie d'exploration du voisinage choisie, un critère d'arrêt naturel consiste à poursuivre la recherche jusqu'à ce que toutes les solutions de l'archive soient marquées comme visitées. En effet, dans un tel cas, l'archive ne contient plus que des solutions non-dominées pour lesquelles tous les voisins sont dominés ou équivalent à au moins une solution de la même archive. L'algorithme est tombé dans une sorte d'optimum local Pareto.

2.1.5.3 Algorithmes existants comme instances du modèle

Le but de cette section est de montrer comment des algorithmes DMLS provenant de la littérature s'inscrivent facilement dans la vue unifiée proposée. Deux recherches locales Pareto :

TABLE 2.2 – Algorithmes de recherche locale multiobjectif existants comme instances du modèle DMLS. Quatre exemples sont présentés : PLS-1 (Paquete *et al.*, 2004), PLS-2 (Talbi *et al.*, 2001; Basseur *et al.*, 2003; Angel *et al.*, 2004), PAES (Knowles et Corne, 2000), and moRBC($|A| : 1+1)^A$ (Aguirre et Tanaka, 2005).

composants	PLS-1	PLS-2	PAES	moRBC$(\|A\| : 1+1)^A$
relation de dominance	dominance Pareto	dominance Pareto	dominance Pareto	dominance Pareto
sélection de l'ensemble courant	*partielle* 1 solution aléatoire	*exhaustive* toutes les solutions	*partielle* μ solutions	*partielle* 1 solution diversifiée
exploration du voisinage	*exhaustive* tous les voisins	*exhaustive* tous les voisins	*partielle* λ voisins aléatoires	*partielle* 1 voisin dominant
gestion de l'archive	*non-bornée*	*non-bornée*	*bornée* hyper-grille	*bornée* distance de *crowding*
condition d'arrêt	*naturelle* solutions visitées	*naturelle* solutions visitées	*définie par l'utilisateur*	*naturelle* solutions visitées

PLS-1 (Paquete *et al.*, 2004), et PLS-2 (Talbi *et al.*, 2001; Basseur *et al.*, 2003; Angel *et al.*, 2004), ainsi que PAES (Knowles et Corne, 2000) et moRBC$(|A| : 1+1)^A$ (Aguirre et Tanaka, 2005) sont pris comme exemples. Le tableau 2.1.5.3 les présente comme de simples instances du modèle DMLS. Naturellement, seuls les composants indépendant du problème sont présentés. Ces méthodes utilisent toutes la relation de dominance Pareto dans leur version originale. Comme illustré sur la figure 2.1.5.3, ces quatre algorithmes DMLS classiques sont basés sur de simples variantes des composants indépendants du problème présenté précédemment, ce qui valide la grande flexibilité et modularité du modèle d'unification proposé.

Exemple 2.5 (PLS-1) (Paquete *et al.*, 2004)
À chaque itération, PLS-1 sélectionne une solution non-visitée au hasard à partir de l'archive, et explore son voisinage de façon exhaustive. Il contient une condition d'arrêt naturelle qui est vérifiée lorsque tous les membres de l'archive sont marqués comme visités. Une archive non-bornée est généralement utilisée, même si un mécanisme de limitation de la taille peut très facilement y être inclus. L'algorithme 5 donne le modèle de PLS-1.

Exemple 2.6 (PLS-2) (Talbi *et al.*, 2001; Basseur *et al.*, 2003; Angel *et al.*, 2004)
La seule différence entre PLS-1 et PLS-2 est que ce dernier algorithme ajoute, à chaque itération, l'ensemble des solutions non-visitées de l'archive à l'ensemble courant, de sorte qu'un ensemble candidat de très grande taille peut potentiellement être construit.

Exemple 2.7 (PAES) (Knowles et Corne, 2000)
À chaque itération de l'algorithme PAES, μ solutions sont sélectionnées à partir de l'archive.

Algorithme 5 Modèle de l'algorithme PLS-1

1. **Initialisation.** Initialiser l'archive A avec une solution $x \in X$ fournie en paramètre, ou la générer de façon aléatoire.
2. **Sélection de l'ensemble courant.** Sélectionner une solution marquée comme non-visitée x aléatoirement parmi les membres de l'archive A.
3. **Exploration du voisinage.** Générer la totalité des voisins de x et ajoutez les solutions non-dominées explorées à l'ensemble candidat $P_{candidat}$. Marquer x comme visitée.
4. **Mise à jour de l'archive.** Mettre à jour l'archive A à l'aide des solutions de l'ensemble candidat $P_{candidat}$.
5. **Condition d'arrêt.** Si les solutions de l'archive A sont toutes marquées comme visitées ou si une autre condition d'arrêt est satisfaite, retourner les solutions non-dominées de l'archive A. Stop. Sinon, aller à l'étape 2.

Une exploration partielle du voisinage est ensuite effectuée à hauteur de λ voisins aléatoires par solution. Dans la version de base de PAES, notez que μ et λ sont tous les deux fixés à 1. La stratégie de sélection de l'ensemble courant consiste à choisir, parmi la solution courante de l'itération précédente et la dernière solution produite, celle qui se trouve dans la région la plus diversifiée de l'archive. Par ailleurs, PAES utilise un technique d'archive bornée basée sur une hyper-grille (voir la section 2.1.3.2). L'algorithme 6 donne la trame générale d'un algorithme PAES pour lequel les paramètres μ et λ sont tous les deux fixés à 1.

Algorithme 6 Modèle de l'algorithme (1+1)-PAES

1. **Initialisation.** Initialiser l'archive A avec une solution $x \in X$ fournie en paramètre, ou la générer de façon aléatoire.
2. **Sélection de l'ensemble courant.** Sélectionner une solution x parmi les membres de l'archive A en fonction de l'état actuel de l'archive, de la solution courante et de la solution candidate de l'itération précédente.
3. **Exploration du voisinage.** Générer un voisin aléatoire x' de x.
4. **Mise à jour de l'archive.** Mettre à jour l'archive A à l'aide de la solution x' à l'aide de la stratégie d'archivage basée sur une hyper-grille.
5. **Condition d'arrêt.** Si une condition d'arrêt est satisfaite, retourner les solutions non-dominées de l'archive A. Stop. Sinon, aller à l'étape 2.

Exemple 2.8 (moRBC($|A| : 1 + 1)^A$) (Aguirre et Tanaka, 2005)
L'algorithme moRBC$(|A| : 1 + 1)^A$ a été originalement proposé pour le cas particulier de problèmes où les solutions sont représentées par un vecteur de bits, et où l'opérateur de voisinage se base sur une modification à l'inverse des valeurs de bits. Néanmoins, il peut être généralisé à tout type de problème et tout type de voisinage. Cet algorithme utilise également une archive bornée à l'aide d'une stratégie de préservation de la diversité basée sur la distance de *crowding*. Une seule solution, celle avec la meilleure distance de *crowding*, est sélectionnée dans l'archive. Ensuite, ses voisins sont évalués jusqu'à ce qu'une solution qui domine la solution courante soit

trouvée, ou une fois que le voisinage correspondant a été totalement exploré. Chaque voisin évalué non-dominé est ajouté à l'ensemble candidat.

2.1.5.4 Un autre exemple d'algorithme de recherche locale multiobjectif

Jusqu'à présent, nous avons supposé que l'ensemble principal de solutions sur lequel se basait la recherche et l'exploration de nouvelles solutions était l'archive. Néanmoins, pour certains problèmes d'optimisation multiobjectif, le nombre de solutions non-dominées peut croître de façon exponentielle par rapport à la taille de l'instance à résoudre. Certains problèmes d'optimisation continus possèdent même un nombre infini de solutions non-dominées. Par conséquent, il se peut qu'une archive de taille non-bornée puisse contenir un très grand nombre de solutions non-dominées, et que la recherche s'avère donc inefficace en explorant le voisinage d'un tel ensemble. Ainsi, un mécanisme additionnel se doit parfois d'être conçu pour contrôler le nombre de solutions à explorer. Deux stratégies sont possibles. La taille de l'archive peut par exemple être bornée, auquel cas la méthode correspondante peut toujours être vue comme une instance du modèle DMLS. Au contraire, une autre stratégie consiste à utiliser une population principale de taille fixe, tout comme dans un algorithme évolutionnaire standard, et n'implique donc pas l'archive dans le processus de recherche.

D'autre part, tout comme nous l'avons vu dans la section 2.1.3.1, le critère d'acceptation d'une recherche locale peut aussi bien se baser sur un indicateur de qualité arbitraire donné en paramètre. L'algorithme IBMOLS (*indicator based multiobjective local search*), proposé par Basseur et Burke (2007), est une méthode de recherche locale à base de population fondée sur un indicateur binaire de qualité. En d'autres mots, il s'agit d'une adaptation de l'algorithme évolutionnaire IBEA (Zitzler et Künzli, 2004) en un algorithme de recherche locale.

L'algorithme IBMOLS maintient une population principale qui peut être initialisée aléatoirement. Ensuite, il génère le voisinage d'une solution de la population jusqu'à ce qu'une solution améliorante, en termes de valeur de fitness et ceci par rapport à toutes les solutions de la population, soit trouvée ; ou encore jusqu'à ce que tous les voisins aient été évalués. Si une solution améliorante est trouvée, elle remplace la moins bonne solution de la population. En itérant ce principe simple à toutes les solutions de la population, nous obtenons une étape de recherche locale. La procédure complète de recherche locale s'arrête lorsque l'archive n'a reçu aucun nouvel élément non-dominé lors d'une étape complète de recherche locale. La méthode IBMOLS possède donc un critère d'arrêt naturel. Les étapes de la méthode IBMOLS sont détaillées dans l'algorithme 7.

2.2 Implémentation

Maintenant que les détails de conception de métaheuristiques pour l'optimisation multiobjectif ont été présentés, cette section se concentre sur les aspects liés à l'implémentation, et dès lors, au transfert technologique des questions étudiées ci-dessus. Les motivations de l'utilisation d'une plateforme logicielle sont détaillées. Ensuite, les principales caractéristiques de la plateforme proposée, ParadisEO-MOEO, sont décrites, et les intérêts d'un tel outil, aussi bien pratiques que théoriques, sont discutés.

Algorithme 7 Modèle de l'algorithme IBMOLS

1. **Initialisation.** Démarrer avec une population initiale P de taille N fournie en paramètre, ou la générer de façon aléatoire ; initialiser l'archive A avec les solutions non-dominées de la population P.
2. **Affectation d'une valeur de fitness.** Calculer les valeurs de fitness pour chaque solution x de la population P :

$$F(x) \leftarrow \sum_{x' \in P \setminus \{x\}} -e^{-I(x',x)/\kappa} \tag{2.9}$$

3. **Étape de recherche locale.** Répéter les étapes suivantes pour chaque solution x de la population courante P.
   (a) $x^\star \leftarrow$ un voisin aléatoire de x.
   (b) Calculer la valeur de fitness de $x^\star$:

$$F(x^\star) \leftarrow \sum_{x' \in P} -e^{-I(x',x^\star)/\kappa} \tag{2.10}$$

   (c) Mettre à jour les valeurs de fitness des solutions de la population. Pour tout $x \in P$:

$$F(x) \leftarrow F(x) - e^{-I(x^\star,x)/\kappa} \tag{2.11}$$

   (d) Ajouter $x^\star$ à la population P.
   (e) $w \leftarrow$ solution de P ayant la moins bonne valeur de fitness.
   (f) Supprimer w de la population P.
   (g) Mettre à jour les valeurs de fitness des solutions restantes. Pour tout $x \in P$:

$$F(x) \leftarrow F(x) + e^{-I(w,x)/\kappa} \tag{2.12}$$

4. **Mise à jour de l'archive.** Mettre à jour l'archive A à l'aide des solutions de la population P.
5. **Condition d'arrêt.** Si l'archive A n'a pas été modifiée depuis l'itération précédente, retourner les solutions de A. Stop. Sinon, aller à l'étape 3.

2.2.1 Motivations

Dans la pratique, il existe une grande diversité de problèmes d'optimisation à résoudre, engendrant un grand nombre de modèles possibles à traiter dans le cadre de méthodes de résolutions basées sur les métaheuristiques. En outre, un nombre croissant de méthodes de recherche générales sont proposées dans la littérature, avec une évolution vers des mécanismes de plus en plus complexes. D'un point de vue pratique, il existe une forte demande de fournir un ensemble d'implémentations de métaheuristiques prêtes à l'emploi, permettant un effort minimum de programmation. D'autre part, un expert tient généralement à concevoir de nouveaux algorithmes, à intégrer de nouveaux composants dans une méthode existante, ou même à combiner les différents mécanismes de recherche. En outre, il est également utile de pouvoir évaluer et comparer différents algorithmes de façon équitable.
Ainsi, comme l'ont fait observer Cahon *et al.* (2004) et Talbi (2009), trois approches principales peuvent être identifiées pour le développement de métaheuristiques : en partant de rien (c'est-à-dire aucune réutilisabilité), réutilisation du code et réutilisation de la conception et du code.

- Tout d'abord, les programmeurs sont tentés de développer et d'implémenter leur propre code à partir de rien. Toutefois, ceci exige du temps et de l'énergie et le code résultant est généralement sujet à de nombreuses erreurs et difficile à maintenir et à faire évoluer.
- La deuxième approche consiste à réutiliser une implémentation de tierce-partie, sous la forme de programmes individuels ou de librairies généralement disponibles sur le web. Les programmes individuels sont souvent composés de sections dépendantes de l'application qui doivent être extraites avant d'insérer le code relatif à l'application traitée. De même, modifier ces sections est souvent coûteux en temps et sujet à erreurs. La réutilisation de code par le biais des bibliothèques est évidemment plus recommandée, car ces dernières sont souvent testées, documentées, et donc plus fiables. Cependant, les bibliothèques ne permettent pas la réutilisation de la partie invariante des algorithmes liés à la conception. Par conséquent, l'effort d'implémentation reste important.
- Enfin, la réutilisabilité du code et de la conception permet de surmonter ce problème. En conséquence, une approche approuvée pour le développement de métaheuristiques est l'utilisation de plateformes logicielles.

Une plateforme logicielle pour la conception et l'implémentation de métaheuristiques peut être définie par un ensemble de composants basés sur une séparation conceptuelle claire de la partie invariante et de la partie spécifique au problème de la métaheuristique. Ainsi, à chaque fois qu'un nouveau problème d'optimisation doit être abordé, le code et la conception peuvent tous deux être directement réutilisés afin de ne développer que la plus petite quantité de code possible. Par conséquent, l'effort d'implémentation est minimum relativement au problème étudié. Généralement, la plateforme logicielle garantie le contrôle total de la partie invariante des algorithmes, celle-ci étant encapsulée dans des squelettes génériques et abstraits qui sont fournis. L'utilisateur n'a donc plus qu'à définir le code relatif aux sections dépendantes du problème. Ainsi, cette partie variable est fixée par la plateforme, mais doit être fournie par l'utilisateur. Pour ce faire, la conception de la plateforme logicielle doit être fondée sur une nette séparation conceptuelle entre les méthodes de résolution et le problème à résoudre. La conception et la programmation orientée-objet sont fortement recommandées dans un tel cas. Une autre façon d'effectuer cette séparation serait de fournir un ensemble de modules pour chaque partie, et de les faire communiquer à l'aide de fichiers texte. Toutefois, cette approche est bien moins flexible que l'approche orientée-objet, et est bien plus coûteuse en temps de calcul. Par ailleurs, notons que deux types

de plateforme logicielle peuvent être distingués : les plateformes dites « boîte blanche » et les plateformes « boîte noire ». Ces dernières réutilisent les composants et se basent sur le couplage par composition et paramétrage statiques. À l'opposé, les plateformes « boîte blanche » requièrent une compréhension des objets des classes correspondantes et une maîtrise à l'utilisation, pour ensuite être en mesure d'implémenter des sous-classes valides par héritage ou par spécialisation.

2.2.2 Présentation de ParadisEO-MOEO

Cette section contient une présentation générale de ParadisEO, une plateforme logicielle dédiée à la conception de métaheuristiques, et une description détaillée de ParadisEO-MOEO, le module de ParadisEO spécifiquement dédié à l'optimisation multiobjectif. Historiquement, ParadisEO était principalement dédié aux métaheuristiques parallèles et distribuées, et était le résultat du travail de thèse de Sébastien Cahon, supervisé par Nouredine Melab et El-Ghazali Talbi (Cahon *et al.*, 2004). La version initiale contenait déjà un petit nombre de fonctionnalités liées à l'optimisation multiobjectif, principalement en ce qui concerne la gestion de l'archive. Ce travail a ensuite été élargi et présenté dans (Liefooghe *et al.*, 2007b). Mais depuis, nous avons presque intégralement remanié le module ParadisEO-MOEO afin de fournir une décomposition plus fine, en adéquation avec le modèle unifié présenté dans la section précédente.

2.2.2.1 La plateforme ParadisEO et le module ParadisEO-MOEO

ParadisEO [3] est une plateforme logicielle « boîte blanche » orientée-objet dédiée à la conception flexible des métaheuristiques pour la résolution de problèmes d'optimisation de nature continue, discrète ou combinatoire. Basé sur la librairie EO (*Evolving Objects*) [4] (Keijzer *et al.*, 2001), ParadisEO s'appuie sur la notion de Template C++, et est portable sur les systèmes Unix (Linux, MacOS) et Windows. Ce logiciel est régi par la licence CeCILL soumise au droit français et respectant les règles de diffusion des logiciels libres [5]. Il tend à être utilisé à la fois par des non-spécialistes et des experts de l'optimisation. Comme illustré dans la figure 2.16, il est composé de quatre modules inter-connectés qui constituent une plateforme globale. Chaque module est fondé sur une séparation conceptuelle claire entre les méthodes de résolution et les problèmes qu'elles sont destinées à résoudre. Cette séparation confère une réutilisation maximum de code et de conception pour l'utilisateur.
Les modules constituant la plateforme ParadisEO sont les suivants.

1. **ParadisEO-EO** (*Evolving Objects*) pour l'implémentation de métaheuristiques à base de population, comprenant entre autres les algorithmes évolutionnaires (Keijzer *et al.*, 2001) et les algorithmes à essaim de particules (Talbi, 2009).
2. **ParadisEO-MO** (*Moving Objects*) (Boisson *et al.*, 2009) pour l'implémentation de métaheuristiques à base de solution unique, c'est-à-dire les méthodes de recherche locale, de recuit simulé, de recherche tabou, etc.
3. **ParadisEO-MOEO** (*Multi-Objective Evolving Objects*) (Liefooghe *et al.*, 2009a,c) pour l'implémentation de métaheuristiques dédiées à l'optimisation multiobjectif.

3. ParadisEO est disponible à l'URL : `http://paradiseo.gforge.inria.fr`.
4. EO est disponible à l'URL : `http://eodev.sourceforge.net`.
5. Des informations sur la licence CeCILL sont disponibles à l'URL : `http://www.cecill.info`.

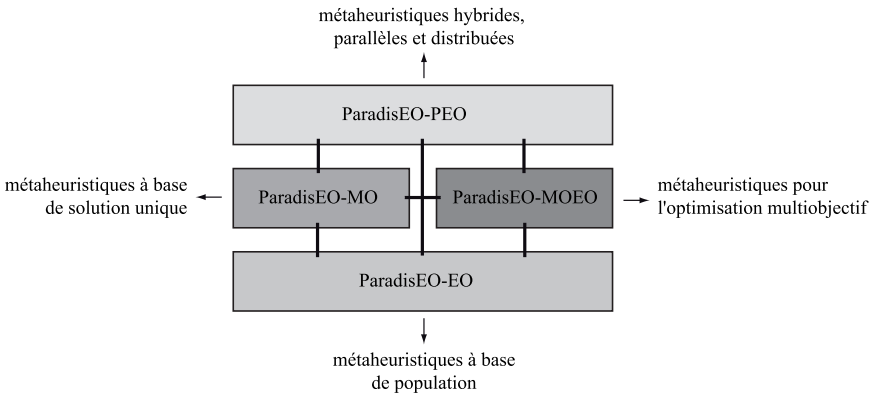


Fig. 2.16 – Les modules de la plateforme ParadisEO.

4. **ParadisEO-PEO** (*Parallel Evolving Objects*) (Cahon *et al.*, 2004; Talbi, 2009) pour l'implémentation de métaheuristiques parallèles et distribuées, que ce soit au niveau algorithmique (*multi-start*, modèle en îles), itératif (parallélisation de l'évaluation des solutions de la population) ou de la solution (parallélisation de la fonction d'évaluation).

Par ailleurs, la plateforme globale ParadisEO offre une vaste gamme de composants pour l'implémentation de métaheuristiques hybrides, grâce à la connexion existante entre ses différents modules. Par la suite, nous allons exclusivement nous concentrer sur le module consacré à l'optimisation multiobjectif, à savoir ParadisEO-MOEO.
ParadisEO-MOEO constitue une plateforme flexible et modulaire pour la conception et l'implémentation de métaheuristiques pour l'optimisation multiobjectif. Elle est basée sur le modèle unifié proposé dans la section précédente et est conceptuellement divisé en composants à grain fin. A chaque niveau de son architecture, un ensemble de classes abstraites est proposé et un large éventail de classes instanciables, correspondant à différentes stratégies classiques de la littérature, est également fourni. En outre, la plateforme a pour but d'être extensible, souple et facilement adaptable, tous ses composants sont génériques afin que son architecture modulaire permette de développer rapidement et facilement toute nouvelle technique, ceci avec un minimum de lignes de code à écrire. L'objectif ici est de suivre les nouvelles stratégies à venir de la littérature et, si besoin est, de fournir des éléments supplémentaires nécessaires à leur mise en œuvre. ParadisEO-MOEO évolue constamment et de nouvelles fonctionnalités sont constamment susceptibles d'être ajoutées afin de fournir les concepts les plus modernes et efficaces, et de tenir compte des avancées les plus récentes du domaine.

2.2.2.2 Caractéristiques principales

Une plateforme logicielle est en général destinée à être exploitée par un grand nombre d'utilisateurs. Son exploitation ne peut être efficace que si un ensemble de critères sont satisfaits. Par conséquent, les principaux objectifs de la plateforme ParadisEO sont les suivants (Cahon *et al.*, 2004; Talbi, 2009).

– **Réutilisabilité maximum de la conception et du code.** La plateforme doit fournir une architecture de conception complète pour la métaheuristique considérée. L'utilisateur ne doit écrire que le minimum de code spécifique à son problème et le processus de développement

doit être fait de façon incrémentale, ceci afin de considérablement simplifier la mise en œuvre et d'alléger le temps de développement, et donc le coût. Cet objectif nécessite une séparation conceptuelle maximale entre les méthodes de résolution et le problème à résoudre.
- **Flexibilité et adaptabilité.** Il doit être possible de facilement ajouter de nouvelles fonctionnalités ou de modifier les fonctionnalités existantes, sans pour autant qu'il y ait de répercussions sur les autres composants. L'utilisateur doit avoir accès au code source et doit utiliser les concepts d'héritage ou de spécialisation de la programmation orientée-objet pour créer de nouveaux composants à partir de composants de base ou de classes abstraites. Par ailleurs, de nouveaux problèmes d'optimisation surgissent et certains problèmes d'optimisation existants se développent, les composants doivent donc être adaptés et spécialisés de façon appropriée.
- **Utilité.** La plateforme doit couvrir un large éventail de métaheuristiques, de composants, des modélisations de problèmes classiques, des modèles parallèles et distribués, des mécanismes d'hybridation, etc. Bien sûr, une utilisation avancée ne doit pas ajouter de difficultés pour les utilisateurs désireux d'utiliser les implémentations basiques.
- **Accès transparent à la performance et à la robustesse.** Étant donné que les méthodes d'optimisation sont souvent coûteuses en temps, en particulier pour les problèmes réels difficiles, la question de performance des métaheuristiques proposées est cruciale. Aussi, le calcul parallèle et distribué est une façon intéressante d'atteindre un niveau élevé de performance à l'exécution. Par ailleurs, l'exécution des algorithmes doit être robuste afin de garantir la fiabilité et la qualité des résultats. Lorsqu'ils sont conçus de manière astucieuse, les mécanismes d'hybridation permettent souvent d'obtenir de solides et de meilleures solutions.
- **Portabilité.** Afin de satisfaire le plus grand nombre d'utilisateurs, la plateforme doit supporter un grand nombre d'architectures matérielles (séquentielle, parallèle, distribuée) et leurs systèmes d'exploitation associés (Windows, Linux, MacOS).
- **Facilité d'utilisation et efficacité.** La plateforme doit être facile d'utilisation et ne doit contenir aucun coût supplémentaire en termes de complexité en temps ou en espace, ceci afin de conserver une efficacité similaire à une implémentation spécifique. Au contraire, celle-ci est destinée à être moins sujette aux erreurs que les métaheuristiques développées de façon spécifique.

ParadisEO honore tous les critères mentionnés ci-dessus et vise à être utilisée à la fois par des non-spécialistes et des experts de l'optimisation. Par ailleurs, le module ParadisEO-MOEO doit couvrir d'autres besoins liés à l'optimisation multiobjectif. Par exemple, il doit pouvoir être très simple d'étendre un problème d'optimisation monoobjectif à sa contrepartie multiobjectif sans pour autant modifier l'ensemble de l'implémentation des composants liés au problème.

2.2.2.3 Plateformes logicielles existantes

Un nombre relativement élevé de plateformes logicielles dédiées à la conception de métaheuristiques ont été proposées jusqu'à présent. Cependant, la plupart d'entre elles sont uniquement dédiées aux algorithmes évolutionnaires, et très peu sont capables de résoudre des problèmes multiobjectif. Quelques unes fournissent tout de même des éléments pour un sous-ensemble particulier des stratégies, parmi lesquelles on peut citer ECJ [6], JavaEVA (Streichert et Ulmer, 2005) ou Open BEAGLE (Gagné et Parizeau, 2006).
Le tableau 2.3 donne une liste non exhaustive de comparaison entre un certain nombre de pla-

6. `http://cs.gmu.edu/~eclab/projects/ecj/`

TABLE 2.3 – Caractéristiques principales des plateformes logicielles existantes pour le développement de métaheuristiques pour l'optimisation multiobjectif.

plateforme	problème		outils stat.		hybrid.	para.	boîte	lang.	licence
	cont.	comb.	off-line	on-line					
jMetal	oui	oui	oui	non	oui	non	blanche	java	libre
MOEA for Matlab	oui	non	non	non	non	oui	noire	matlab	comm.
MOMHLib++	oui	oui	non	non	oui	non	blanche	c++	libre
PISA	oui	oui	oui	non	non	non	noire	-	libre
Shark	oui	non	non	non	oui	non	blanche	c++	libre
ParadisEO	oui	oui	oui	oui	oui	oui	blanche	c++	libre

teformes logicielles existantes pour l'implémentation de métaheuristiques pour l'optimisation multiobjectif. Sont incluses les plateformes jMetal (Durillo *et al.*, 2006), la boîte à outils MOEA pour Matlab (Tan *et al.*, 2001), MOMHLib++ [7], PISA (Bleuler *et al.*, 2003) et Shark (Igel *et al.*, 2008). Notez que d'autres logiciels existent pour l'optimisation multiobjectif (Poles *et al.*, 2008), mais certains ne peuvent pas être considérés comme des plateformes et d'autres ne traitent pas de métaheuristiques. Les plateformes présentées dans le tableau sont distinguées les unes des autres selon les critères suivants : le genre de problème qu'elles sont capables de traiter : continu (cont.) et/ou combinatoire (comb.), la disponibilité d'outils statistiques (y compris les indicateurs de performance) en dehors du processus de recherche (off-line) et durant le processus de recherche (on-line), la disponibilité de modèles parallèles (para.) ou d'hybridation (hybrid.), le type de plateforme (boîte noire ou boîte blanche), le langage de programmation (lang.) et le type de licence (libre ou commercial).

Tout d'abord, laissez-nous mentionner que toutes les plateformes logicielles répertoriées sont libre d'utilisation, à l'exception de la boîte à outils MOEA conçue pour logiciel commercial Matlab. Elles sont toutes en mesure de gérer des problèmes continus, mais seulement un sous-ensemble peut faire face à des problèmes multiobjectif combinatoires. Par ailleurs, notez que certaines d'entre elles ne peuvent pas être considérées comme des plateformes boîte-blanche puisque leur architecture n'est pas décomposée en différents composants. Par exemple, pour concevoir une nouvelle méthodologie sous PISA, il est quasiment nécessaire de l'implémenter à partir de rien, car aucun élément existant ne peut être réutilisé. Aussi, même si Shark peut être considéré comme une plateforme boîte-blanche, ses composants ne sont pas aussi fins que ceux de ParadisEO. Au contraire, ParadisEO est une plateforme ouverte où chacun peut contribuer et ajouter ses propres fonctionnalités. Enfin, seul un petit nombre d'entre elles sont en mesure de concevoir à la fois des métaheuristiques parallèles et hybrides. Ainsi, par rapport à la taxinomie proposée par Talbi (2002), seule la classe de métaheuristiques hybrides en mode relais peuvent être facilement implémenter sous jMetal, MOMHLib++ et Shark, alors que ParadisEO fournit les outils nécessaires à la conception de toutes les classes de modèles hybrides, dont les métaheuristiques hybrides en mode *teamwork*. De plus, en opposition à jMetal et MOMHLib++, ParadisEO propose des modèles faciles à utiliser pour la conception de modèles parallèles et distribués de métaheuristiques pour l'optimisation multiobjectif. Par conséquent, ParadisEO semble être la seule plateforme logicielle existante qui satisfasse tous les critères susmentionnés.

7. `http://home.gna.org/momh/`

2.2.3 Implémentation sous ParadisEO-MOEO

Les détails techniques d'implémentation de métaheuristiques pour l'optimisation multiobjectif sous ParadisEO-MOEO sont donnés en annexe. La haute flexibilité de la plateforme et son architecture modulaire, basée sur les trois notions principales spécifiques à la conception de métaheuristiques pour l'optimisation multiobjectif (affectation d'une valeur de fitness, préservation de la diversité et élitisme) permettent d'implémenter des algorithmes évolutionnaires et des algorithmes de recherche locale efficaces afin de résoudre une grande variété de problèmes d'optimisation multiobjectif. La décomposition granulaire de ParadisEO-MOEO est basée sur les modèles d'unification proposés dans la section précédente.
Ainsi, plusieurs classes fournies au sein de ParadisEO-MOEO permettent de définir un algorithme évolutionnaire ou un algorithme DMLS de façon générique, par simple instanciation. Différents composants peuvent donc être très facilement expérimentés sans pour autant engendrer d'importantes modifications en termes d'écriture de code. Mais, il faut garder à l'esprit que cette liste n'est pas exhaustive, car la plateforme est en perpétuelle évolution et offre tout ce qui est nécessaire pour développer de nouveaux composants avec un minimum d'effort.
Par ailleurs, de nombreuses implémentations d'algorithmes classiques, basées sur une combinaison de composants fournis au sein de la plateforme, sont proposées, parmi lesquelles les algorithmes évolutionnaires MOGA (Fonseca et Fleming, 1993), NSGA (Srinivas et Deb, 1994), NSGA-II (Deb *et al.*, 2002), SPEA2 (Zitzler *et al.*, 2001b), IBEA (Zitzler et Künzli, 2004) et SEEA (Liefooghe *et al.*, 2008e), ainsi que les algorithmes de recherche locale PLS-1 (Paquete *et al.*, 2004), PLS-2 (Talbi *et al.*, 2001; Basseur *et al.*, 2003; Angel *et al.*, 2004), et IBMOLS (Basseur et Burke, 2007). Au sein de ces implémentations, un nombre minimum de paramètres spécifiques à l'algorithme ou au problème à résoudre sont nécessaires. Par exemple, pour instancier NSGA-II pour un problème multiobjectif continu original, il est possible d'utiliser les opérateurs de représentation, d'initialisation et de variation fournis, l'évaluation étant donc le seul composant à implémenter. Il en va de même pour n'importe quel autre de ces algorithmes et pour n'importe quel problème pouvant être représenté à l'aide d'un vecteur de bits, d'entiers ou encore d'une permutation. Ces algorithmes faciles à utiliser tendent à être utilisés comme références dans le monde académique, ceci afin de pouvoir équitablement comparer les performances de différentes méthodes, leur implémentation partageant les mêmes composants de base. Néanmoins, elles se montrent tout autant adaptées à une application directe en vue de la résolution d'un problème d'optimisation multiobjectif issu du monde réel.

2.2.4 Discussion

Nous pensons que les caractéristiques susmentionnées font de ParadisEO-MOEO un outil précieux à la fois pour les chercheurs et les praticiens, et une plateforme logicielle compétitive par rapport à l'existant. En effet, elle comprend un nombre important de métaheuristiques classiques et modernes pour l'optimisation multiobjectif, ainsi qu'un ensemble important de composants modulaires formant les éléments de base de l'implémentation de ces méthodes. De plus, ParadisEO offre la possibilité de concevoir et d'implémenter un nombre important de nouvelles méthodes de résolution de problèmes multiobjectif, soit en combinant les composants existants de manière novatrice, soit en en implémentant aisément de nouveaux. Par ailleurs, le code source est maintenu et régulièrement mis à jour par les développeurs ; de nouvelles méthodologies de résolution seront probablement régulièrement proposées au sein de la plateforme. À titre d'exemple, pour

illustrer la grande flexibilité de ParadisEO, la relation de dominance utilisée pour gérer l'archive d'un algorithme est juste un paramètre de ce composant. Ce dernier peut alors être modifié à partir de, par exemple, une relation de dominance Pareto classique vers une relation d'ϵ-dominance avec un effort négligeable, et a pour effet de concevoir une méthode de résolution grandement différente. En outre, par analogie à l'aspect conceptuel, un grand nombre des composants est également partagé par de nombreux algorithmes en termes d'implémentation. Par conséquent, nous pensons que ParadisEO est un candidat sérieux pour servir d'implémentation de référence lors de la comparaison de différents algorithmes de façon équitable. Par exemple, quand une nouvelle métaheuristique est proposée, son efficacité peut être démontrée expérimentalement en comparant son comportement à des algorithmes existants au sein de la plateforme, et ceci sur un ensemble de problèmes différents.

Métaheuristiques hybrides et parallèles. La conception et l'implémentation de métaheuristiques séquentielles ne sont qu'un aspect des fonctionnalités offertes par ParadisEO. En effet, les algorithmes présentés peuvent facilement être hybridés, parallélisés et distribués grâce au module ParadisEO-PEO (Talbi, 2009). La plateforme a été conçue de telle manière que l'utilisateur peut, par exemple, paralléliser un algorithme évolutionnaire multiobjectif séquentiel très facilement et de manière transparente. Lors de la résolution de problèmes d'optimisation réels, exécuter un algorithme séquentiel sur une large population ou sur des solutions complexes nécessite souvent de hautes ressources de calcul. Toutefois, étant donné que chaque membre de la population peut être considéré comme une unité indépendante, le parallélisme survient naturellement lorsque l'on traite de métaheuristiques à base de population. Melab *et al.* (2006); Talbi (2009) identifient trois niveaux de parallélisme pour les métaheuristiques. Tout d'abord, au niveau algorithmique, plusieurs métaheuristiques indépendantes ou coopérant les unes avec les autres s'exécutent en parallèle. Au niveau itératif, à chaque itération de la métaheuristique, une ou plusieurs étapes sont exécutées en parallèle telles que l'évaluation de la population ou l'exploration du voisinage d'une solution. Enfin, au niveau de la solution, le processus de parallélisation ne manipule qu'une seule solution afin, par exemple, de paralléliser le calcul de la fonction d'évaluation en elle-même. ParadisEO est l'une des rares plateformes logicielles fournissant ces différents modèles parallèles et distribués. En termes d'implémentation, ParadisEO utilise des bibliothèques standards comme MPI (Message Passing Interface), Pthreads, Condor ou Globus. Par conséquent, les applications sont portables sur différentes plateformes d'exécution telles que des machines multiprocesseurs, des clusters, des grilles de calcul, etc.

Coût de développement. D'autre part, ParadisEO est également un outil pratique qui peut être utilisé pour s'attaquer à un problème original. L'implémentation de programmes de résolution efficaces est fortement facilitée, de sorte que l'utilisateur n'a plus qu'à se concentrer sur les questions liées au problème qu'il a à résoudre : représentation, initialisation, évaluation, etc. De surcroît, le travail d'implémentation à fournir est encore plus réduit quand une représentation classique peut être utilisée pour le problème traité. Par représentation classique, nous incluons les représentations basées sur des valeurs binaires, discrètes, réelles ou encore sur une permutation. Pour résoudre ce type de problème, le coût de développement est réduit au minimum, car la fonction d'évaluation n'est plus que le seul élément à implémenter. Bien sûr, ce coût est, et sera toujours, lié aux compétences du programmeur en charge de l'implémentation. Une fois cette fonction d'évaluation disponible en tant que composant de ParadisEO, l'utilisateur n'a plus

qu'à instancier toute métaheuristique disponible (NSGA-II, SPEA2, IBEA, etc.) et, éventuellement, tout modèle parallèle (évaluation parallèle, modèle en îles, etc.) pour obtenir un puissant programme de résolution ; programme qui est capable de fonctionner sur une large gamme d'architectures matérielles (séquentielle, parallèle, distribué) et leurs systèmes d'exploitation (Windows, Linux, MacOS). Par conséquent, la fonction d'évaluation exclue, le développement se résume à une instanciation directe des composants choisis. Bien que, pour un choix de représentation plus sophistiqué, les coûts de développement restent importants à l'égard de la complexité de l'encodage choisi et du niveau d'expertise du programmeur. Mais ce coût sera toujours inférieur à celui de l'implémentation d'une métaheuristique dédiée à partir de rien. Par ailleurs, en partant d'un problème d'optimisation à un seul objectif implémenté dans ParadisEO, il est insignifiant d'implémenter une variante multiobjectif du même problème. Les algorithmes multiobjectif existants peuvent alors être instanciés pour résoudre le problème résultant.

Efficacité d'exécution. En ce qui concerne l'efficacité d'exécution, il est difficile, sinon impossible, d'étudier le comportement de ParadisEO-MOEO par rapport aux plateformes logicielles existantes, ou même à une implémentation réalisée à partir de rien. En effet, ceci dépendra toujours de la capacité du programmeur à gérer tel ou tel outil, et du problème à résoudre ou de la méthodologie appliquée. Développer à partir de rien permet généralement une grande optimisation du code par rapport à une application très spécifique, mais cela exige un haut niveau d'expertise, et est également élevé en termes de coûts de développement. En outre, ParadisEO-MOEO possède l'avantage clair de ne pas communiquer certaines informations par le biais de fichiers texte, comme cela se fait avec PISA, par exemple ; ce qui est connu pour être très coûteux en temps de calcul.

Applications. ParadisEO-MOEO a déjà été utilisé et expérimenté pour résoudre un large éventail de problèmes d'optimisation multiobjectif provenant à la fois du monde académique et du monde réel, ce qui valide sa grande flexibilité. En effet, divers problèmes académiques ont été abordés, comprenant des ensembles de fonctions de test continues telles que les familles ZDT et DTLZ (Deb *et al.*, 2005b) ou encore le problème introduit par Schaffer (1985), un problème d'ordonnancement de type Flowshop (Liefooghe *et al.*, 2007a), des problèmes de tournées multiobjectif tels que le problème du voyageur de commerce multiobjectif ou le problème de Ring-Star (Liefooghe *et al.*, 2010), etc. En outre, ParadisEO-MOEO a été utilisé avec succès pour résoudre des applications du monde réel dans la biologie structurale (Boisson *et al.*, 2008)[8], la sélection d'attributs pour la classification des tumeurs cancérigènes (Talbi *et al.*, 2008), la conception des matériaux chimiques (Schuetze *et al.*, 2008), la gestion de portefeuille, etc. Certains de ces problèmes seront traités dans la section suivante. Par ailleurs, notez que les composants de certaines des applications mentionnées sont disponibles en tant que contributions sur le site web de ParadisEO, et sont accompagnés d'une documentation détaillée et de tutoriaux. Nous prévoyons également une nette augmentation du nombre de contributions développées pour des problèmes multiobjectif dans un avenir proche. En outre, diverses métaheuristiques hybrides et parallèles ont été conçues à l'aide de ParadisEO pour résoudre des problèmes d'optimisation multiobjectif. Par exemple, des algorithmes hybrides seront présentés et expérimentés dans le

8. Cette implémentation est d'ailleurs utilisée au sein du projet Docking@GRID (`http://dockinggrid.gforge.inria.fr`).

chapitre suivant, un modèle de coopération en île a été conçu par Talbi *et al.* (2007), et une fonction d'évaluation coûteuse a été parallélisée par Boisson *et al.* (2008).

2.3 Analyse expérimentale

La dernière section de ce chapitre est dédiée à l'analyse expérimentale des métaheuristiques présentées jusqu'ici, et à leur application aux problèmes abordés dans cette thèse : le problème de Flowshop et le problème de Ring-Star. Le développement de métaheuristiques efficaces est un processus complexe qui combine les aspects de conception et d'implémentation présentés auparavant, mais aussi d'application et d'analyse expérimentale auxquelles nous allons nous intéresser ci-dessous. Dans un premier temps, nous allons tenter d'analyser le comportement et les interactions existants entre les composants du modèle DMLS à l'aide de la résolution du problème de Flowshop à deux et à trois objectifs. Puis, dans un second, temps, nous allons proposer une modélisation pour le problème de Ring-Star multiobjectif, et étudier comparativement le comportement d'un nombre de métaheuristiques (à la fois des algorithmes évolutionnaires et de recherche locale) adaptées à sa résolution.

2.3.1 Application au problème de Flowshop

Les premières expérimentations qui vont nous intéresser concernent le problème d'ordonnancement de type Flowshop introduit dans la section 1.4.1. Nous allons analyser l'efficacité d'un certain nombre d'algorithmes de recherche locale, basés sur le modèle DMLS, pour résoudre le problème de Flowshop, à la fois sous une variante à deux et à trois objectifs (respectivement dénotées FSP-2 et FSP-3). Notez que le but de cette partie expérimentale n'est pas de fournir la méthode de recherche obtenant les meilleurs résultats possibles pour le cas particulier du problème traité, mais plutôt de discuter des comportements respectifs de ces différentes stratégies. Ainsi, nous essayerons de fournir des lignes directrices à propos des questions essentielles liées à la conception d'une méthode de type DMLS. Cette résolution peut également être vue comme un exemple d'application du modèle DMLS proposé.
Nous allons commencer par présenter les composants spécifiques liés à la modélisation du problème de Flowshop pour une résolution basée sur les métaheuristiques. Ensuite, nous présenterons en détail les approches de résolution étudiées et le protocole d'expérimentation que nous allons utiliser. Enfin, les résultats expérimentaux seront présentés et analysés.

2.3.1.1 Modélisation

Les composants liés à la résolution d'un problème spécifique à l'aide de métaheuristiques basées sur les algorithmes évolutionnaires et les algorithmes de recherche locale concernent la représentation, l'évaluation, l'initialisation, ainsi que l'opérateur de voisinage et l'évaluation incrémentale pour les algorithmes de recherche locale, et les opérateurs de variation (croisement, mutation) pour les algorithmes évolutionnaires. Ceux que nous allons utiliser pour le cas particulier du problème de Flowshop étudié ici sont présentés ci-dessous.

2.3.1.1.1 Représentation. Dans le cadre général, une solution réalisable pour un problème d'ordonnancement de type Flowshop doit être représentée par la séquence ordonnée des jobs

exécutés sur chacune des machines. Néanmoins, le problème que nous considérons ici est un problème de Flowshop de permutation, où les séquences de jobs sont identiques et unidirectionnelles pour chacune des machines. Ainsi, la séquence ordonnée des jobs exécutés sur une seule machine suffit à décrire une solution. Une solution à un problème de taille N (où N est le nombre de jobs) pourra donc représentée par une permutation de taille N.

2.3.1.1.2 Évaluation. L'étape d'évaluation se résume tout simplement au calcul des fonctions objectif liées au problème : le makespan et le retard total pour FSP-2, le makespan, le retard total et le retard maximum pour FSP-3.

2.3.1.1.3 Initialisation. La ou les solutions initiales sont générées à l'aide de permutations aléatoires.

2.3.1.1.4 Voisinage. De nombreux opérateurs de voisinage existent pour des représentations à base de permutations (Schiavinotto et Stützle, 2007). Pour les problèmes d'ordonnancement, deux types d'opérateur peuvent être distingués : les voisinages fondés sur l'ordre des éléments et les voisinages fondés sur la position des éléments. Les premiers, tel que l'opérateur *2-opt*, se montrent peu adaptés à notre problème, pour lequel la position des jobs est plus importante que leur ordre. Ainsi, un opérateur de voisinage, fondé sur la position des éléments, et qui semble approprié pour le problème de Flowshop étudié ici, s'avère être l'opérateur d'insertion. C'est un opérateur couramment utilisé pour notre problème, et qui offre de bons résultats (Basseur, 2005). Il consiste à supprimer un élément situé à une position i, et à le réintroduire à une autre position j, tel qu'illustré sur la figure 2.17. Les jobs situés entre les positions i et j sont donc décalés. Le nombre de voisins par solution est $(N-1)^2$, où N correspond à la taille de la permutation (le nombre de jobs pour l'instance de problème considérée). Notez qu'aucun ordre particulier n'est considéré entre les mouvements associés à une solution ; les voisins sont examinés dans un ordre aléatoire.

2.3.1.1.5 Évaluation incrémentale. Afin d'évaluer de façon incrémentale une solution voisine pour le problème de Flowshop étudié, il est nécessaire de stocker les dates de complétion de chaque job sur chaque machine, et ceci pour chaque solution ; ce qui s'avère rapidement impraticable dès lors qu'une métaheuristique à base de population est développée. Pour cette raison, aucune évaluation incrémentale n'est ici considérée, chaque solution voisine sera évaluée à partir de zéro.

2.3.1.1.6 Variation. Nous présentons ici les opérateurs de croisement et de mutation qui seront utilisés par les algorithmes évolutionnaires appliqués au problème de Flowshop.

Croisement. L'opérateur de croisement que nous allons utiliser est le croisement deux-points proposé par Ishibuchi et Murata (1998). Il consiste à choisir deux positions de croisement aléatoires. Le premier individu Enfant conserve les extrémités du premier individu Parent, et les jobs manquants sont insérés dans l'ordre dans lequel ils apparaissent chez le deuxième individu Parent. Cet opérateur est illustré sur la figure 2.18. Un second individu Enfant est également créé en inversant le rôle des deux individus Parent ; l'opérateur est donc quadratique.

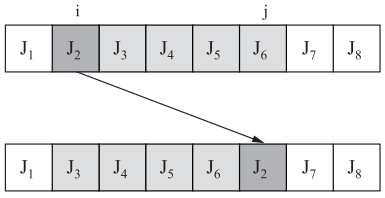


Fig. 2.17 – Illustration de l'opérateur *insertion* (voisinage, mutation) pour le problème de Flowshop.

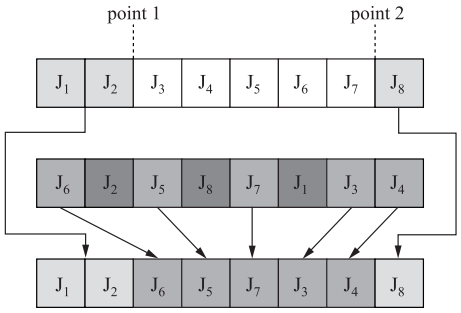


Fig. 2.18 – Illustration de l'opérateur de croisement pour le problème de Flowshop.

Mutation. De façon générale, l'opérateur de mutation d'algorithmes évolutionnaires est apparenté à l'opérateur de voisinage des algorithmes de recherche locale. Ainsi, l'opérateur de mutation que nous allons utiliser pour le problème de Flowshop est basé sur l'opérateur d'insertion présenté précédemment (Fig. 2.17). Il consiste simplement à choisir la position de suppression i et la position de réinsertion j de façon aléatoire.

2.3.1.2 Approches étudiées

Au cours de cette analyse expérimentale, différentes stratégies sont expérimentées pour les étapes de sélection de l'ensemble courant et d'exploration du voisinage des algorithmes DMLS. Cela donne lieu à un ensemble de huit méthodes de recherche locale qui sont résumées dans la figure 2.19. Tout d'abord, en ce qui concerne la sélection de l'ensemble courant, deux stratégies simples sont étudiées, soit (i) une seule solution aléatoire ou (ii) l'ensemble des solutions est sélectionné parmi les solutions non visitées de l'archive. Ensuite, en ce qui concerne les techniques d'exploration du voisinage, trois stratégies d'exploration partielle et une stratégie d'exploration exhaustive sont étudiées.

- **1 voisin aléatoire**. Un seul voisin aléatoire par solution est proposé comme solution candidat pour intégrer l'archive.
- **1er voisin non-dominé**. Pour chaque solution x de l'ensemble courant, le premier voisin trouvé qui est non-dominé par rapport à x est proposé comme solution candidate.
- **1er voisin dominant**. Pour chaque solution x de l'ensemble courant, les voisins sont évalués jusqu'à ce qu'une solution dominant x soit trouvée. Tous les voisins non-dominés évalués sont alors proposés comme candidats pour intégrer l'archive.
- **Tous les voisins**. Tous les voisins de chaque solution de l'ensemble courant sont proposés comme solutions candidates.

Les algorithmes résultants seront dénotés par DMLS $(1 \cdot 1)$, DMLS $(1 \cdot 1_{\not\prec})$, DMLS $(1 \cdot 1_{\succ})$, DMLS $(1 \cdot \star)$, DMLS $(\star \cdot 1)$, DMLS $(\star \cdot 1_{\not\prec})$, DMLS $(\star \cdot 1_{\succ})$, et DMLS $(\star \cdot \star)$, comme indiqué sur la figure. 2.19.

Certains de ces algorithmes correspondent exactement, ou sont étroitement liés à des approches existantes de la littérature. Ainsi, DMLS $(1 \cdot \star)$ est équivalent à PLS-1 (Paquete *et al.*, 2004), et DMLS $(\star \cdot \star)$ à PLS-2 (Talbi *et al.*, 2001; Basseur *et al.*, 2003; Angel *et al.*, 2004). Ensuite, DMLS $(1 \cdot 1)$ est en quelque sorte apparenté à la stratégie (1+1)-PAES proposée par Knowles et Corne (2000). Les principales différences apparaissent lors de l'étape de sélection de l'ensemble

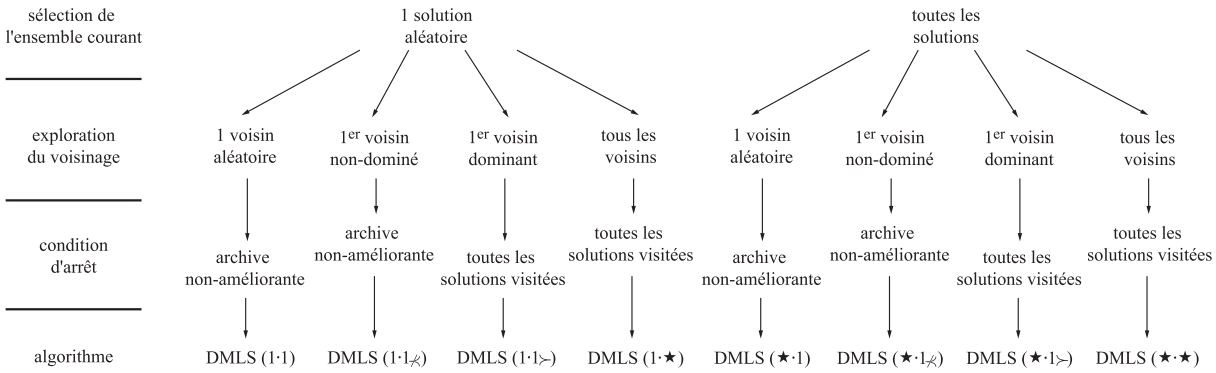


Fig. 2.19 – Algorithmes DMLS étudiés.

courant, où la solution est choisie aléatoirement, et lors de l'étape d'archivage, où aucune technique d'archivage bornée n'est considérée. Enfin, DMLS $(1 \cdot 1_{\succ})$ est étroitement lié à l'algorithme moRBC$(|A| : 1 + 1)^A$ (Aguirre et Tanaka, 2005), mis à part que ce dernier a initialement été proposé pour un problème particulier, et qu'il utilise une technique basée sur le *crowding* pour borner la taille de l'archive et pour sélectionner la solution courante.
Le mécanisme d'archivage utilisé par tous les algorithmes DMLS consiste à utiliser une archive non-bornée basée sur la relation de dominance Pareto classique. Les conditions d'arrêt respectives des algorithmes sont détaillées ci-dessous.

2.3.1.3 Design expérimental

La condition d'arrêt de toutes les méthodes expérimentées est basée sur un temps d'exécution maximum. Or, certaines stratégies possèdent déjà un critère d'arrêt naturel, il s'avère donc nécessaire de définir une condition de reprise. Par ailleurs, notez que les algorithmes ont tous été implémentés avec la version 1.2.1 de ParadisEO, et partagent donc les mêmes composants de base. Toutes les expérimentations ont été menées sur une même machine, dotée d'un processeur Intel Core 2 Duo 6600 (2×2.40 GHz, 2 Go RAM) avec G++ 4.1.2 fonctionnant sous Linux. La comparaison des algorithmes DMLS étudiés sur le problème de Flowshop est donc juste et équitable.

2.3.1.3.1 Conditions d'arrêt et de reprise. Pour chaque instance du problème traité, un temps maximum d'exécution est autorisé, avec une sauvegarde régulière de l'approximation en cours, ceci en vue d'étudier l'évolution de l'efficacité de la recherche au fil du temps. Cependant, il est vrai que certains algorithmes s'arrêtent de manière naturelle. En effet, lorsque le voisinage de tous les membres de l'archive a été exploré de façon exhaustive, les solutions sont marquées comme visitées et l'algorithme s'arrête ; voir la section 2.1.5.2.6. Dans un tel cas, une simple stratégie de reprise aléatoire est effectuée pour continuer le processus de recherche. Par conséquent, la recherche redémarre à l'aide d'une nouvelle population initiale aléatoire. Ce mécanisme est répété autant de fois que nécessaire pour atteindre le temps maximum d'exécution disponible.
Toutefois, afin de ne pas pénaliser les algorithmes qui ne s'arrêtent pas naturellement, une simple condition d'arrêt non améliorante a été conçue. Celle-ci est vérifiée lorsqu'un certain nombre d'itérations consécutives ont été effectuées sans qu'aucune nouvelle solution non-dominée n'intègre

l'archive. Cette valeur est fixée à v pour les stratégies exhaustives de sélection, et à $(v \times |A|)$ pour les stratégies partielles de sélection, où v est le nombre de voisins par solution et $|A|$ est la taille de l'archive courante. Rappelons que pour le problème de Flowshop étudié ici, la taille du voisinage $v = (N-1)^2$, où N représente le nombre de jobs pour l'instance considérée. Si une itération est détectée comme non-améliorante, notez que la taille de l'archive $|A|$ reste constante au fil du temps. Ces valeurs ont donc été motivées par le fait que, durant un tel nombre d'itérations, les algorithmes correspondants auraient eu le temps de générer la totalité du voisinage de l'archive courante. La condition d'arrêt de chacun des algorithmes étudiés est indiquée sur la figure 2.19.

2.3.1.3.2 Paramètres. Pour toutes les expérimentations, la taille de la population initiale est fixée à 1. Pour la condition d'arrêt, un temps d'exécution maximum est fixé par rapport à la taille de l'instance considérée. Par conséquent, nous nous sommes fixé un temps maximum d'exécution de :
– 10 minutes pour les instances à 20 jobs ;
– 20 minutes pour les instances à 50 jobs ;
– 30 minutes pour les instances à 100 jobs.
Le nombre de solutions réalisables pour le problème de Flowshop est égal à $N!$, où N est le nombre de jobs de l'instance. La condition d'arrêt est donc définie en fonction de la taille de l'espace de recherche.

2.3.1.4 Résultats expérimentaux

Les résultats obtenus pour FSP-2 et FSP-3 par rapport aux indicateurs hypervolume et epsilon sont respectivement présentés dans les tableaux 2.4 et 2.5. Pour des questions de lisibilité, seuls les résultats statistiques pour trois valeurs différentes de temps d'exécution, fixées en fonction de l'instance considérée, sont présentés : un temps court, moyen, et long.
Tout d'abord, nous remarquons que deux ensembles d'instances peuvent clairement être distingués en fonction des performances algorithmiques. La première série contient des instances de relativement petite taille, comprenant les instances FSP-2 de 20 et 50 jobs ainsi que les instances FSP-3 de 20 jobs. La deuxième contient des instances de grande taille et comprend les instances FSP-2 de 100 jobs et les FSP-3 de 50 et 100 jobs. Bien sûr, en termes de taille, nous parlons ici à la fois de la taille de l'espace de recherche et du nombre de fonctions objectif.
Pour l'ensemble de petite taille, les deux algorithmes les plus performants sont clairement DMLS $(1 \cdot 1_{\succ})$ et DMLS $(\star \cdot 1_{\succ})$. Les seules exceptions sont les instances $(020 \times 10 \times 01)$ et $(050 \times 20 \times 01)$ du problème FSP-2, où DMLS $(1 \cdot 1_{\succ})$ (respectivement DMLS $(\star \cdot 1_{\succ})$) est surpassé par au moins un autre algorithme, incluant DMLS $(\star \cdot 1_{\succ})$ (respectivement DMLS $(1 \cdot 1_{\succ})$), lorsqu'un temps d'exécution court est autorisé. Toutefois, DMLS $(1 \cdot \star)$ semble être un concurrent solide en termes d'hypervolume, même s'il est toujours dominé selon l'indicateur $\mathrm{I}^1_{\epsilon+}$. En outre, les résultats obtenus par DMLS $(\star \cdot \star)$ ne sont pas trop faibles pour les très petites instances du problème FSP-2, mais celui-ci ne peut généralement pas concurrencer les autres méthodes pour les plus grandes instances. Enfin, sauf à quelques exceptions près, toutes les autres méthodes ont globalement obtenu de piètres résultats en comparaison aux méthodes mentionnées ci-dessus. Ainsi, on peut raisonnablement conclure que stopper l'exploration du voisinage à l'aide d'une stratégie fondée sur la relation de dominance de Pareto pour chaque solution de l'ensemble actuel

TABLE 2.4 – Comparaison des algorithmes DMLS selon l'indicateur I_H^-. La première valeur représente le nombre d'algorithmes qui domine significativement l'algorithme considéré. La valeur entre parenthèses correspond à la I_H^--valeur moyenne normalisée ($\times 10^{-1}$).

Instance	Temps d'exécution	DMLS $(1 \cdot 1)$	DMLS $(1 \cdot 1_{\not\succ})$	DMLS $(1 \cdot 1_{\succ})$	DMLS $(1 \cdot \star)$	DMLS $(\star \cdot 1)$	DMLS $(\star \cdot 1_{\not\succ})$	DMLS $(\star \cdot 1_{\succ})$	DMLS $(\star \cdot \star)$
FSP-2									
$020 \times 05 \times 01$	1'	4 (2.455)	6 (6.083)	**0** (0.200)	**0** (0.245)	5 (3.907)	5 (4.914)	**0** (0.119)	3 (0.705)
	5'	4 (0.320)	6 (3.115)	**0** (0.161)	1 (0.030)	6 (1.645)	4 (0.445)	**0** (0.026)	1 (0.032)
	10'	4 (0.155)	6 (1.005)	**0** (0.009)	2 (0.017)	6 (0.906)	4 (0.208)	**0** (0.011)	3 (0.021)
$020 \times 10 \times 01$	1'	4 (3.722)	4 (3.784)	1 (1.765)	1 (1.689)	7 (7.403)	4 (3.666)	**0** (1.030)	1 (1.868)
	5'	4 (1.498)	4 (1.662)	**0** (0.320)	2 (0.861)	7 (3.512)	4 (1.735)	**0** (0.353)	**0** (0.535)
	10'	4 (0.829)	5 (1.206)	**0** (0.138)	2 (0.351)	7 (2.151)	4 (1.075)	**0** (0.141)	2 (0.399)
$020 \times 20 \times 01$	1'	3 (3.786)	4 (4.137)	**0** (1.167)	2 (2.361)	7 (7.345)	2 (3.616)	**0** (1.448)	2 (2.952)
	5'	4 (1.181)	6 (1.992)	**0** (0.372)	1 (0.539)	7 (3.756)	4 (1.261)	**0** (0.449)	2 (0.747)
	10'	4 (0.659)	5 (1.351)	**0** (0.154)	**0** (0.392)	7 (2.593)	4 (0.829)	**0** (0.227)	2 (0.385)
$050 \times 05 \times 01$	2'	2 (1.121)	2 (1.130)	**0** (0.800)	2 (0.992)	4 (1.221)	2 (1.054)	**0** (0.777)	7 (1.817)
	10'	3 (0.740)	6 (0.887)	**0** (0.479)	1 (0.546)	5 (0.878)	4 (0.800)	**0** (0.398)	2 (0.670)
	20'	3 (0.593)	6 (0.799)	**0** (0.290)	**0** (0.358)	5 (0.757)	5 (0.733)	**0** (0.263)	3 (0.506)
$050 \times 10 \times 01$	2'	1 (1.519)	**0** (1.443)	**0** (1.372)	5 (1.849)	5 (1.707)	**0** (1.406)	**0** (1.337)	7 (5.156)
	10'	3 (1.070)	4 (1.296)	**0** (0.603)	2 (9.270)	3 (1.157)	4 (1.316)	**0** (0.652)	2 (1.062)
	20'	3 (0.876)	6 (1.271)	**0** (0.467)	2 (0.611)	5 (1.014)	5 (1.187)	**0** (0.508)	3 (0.758)
$050 \times 20 \times 01$	2'	**0** (1.225)	**0** (1.274)	**0** (1.302)	5 (1.794)	2 (1.413)	**0** (1.220)	5 (1.698)	7 (5.864)
	10'	**0** (0.783)	5 (1.069)	**0** (0.666)	**0** (0.781)	3 (0.909)	2 (0.964)	**0** (0.732)	7 (1.644)
	20'	2 (0.652)	6 (1.060)	**0** (0.477)	**0** (0.564)	4 (0.810)	4 (0.937)	**0** (0.476)	2 (0.739)
$100 \times 10 \times 01$	3'	4 (1.164)	**0** (0.601)	1 (0.790)	6 (3.638)	4 (1.125)	**0** (0.700)	2 (0.874)	7 (7.961)
	15'	1 (0.484)	**0** (0.285)	1 (0.431)	6 (1.277)	1 (0.463)	1 (0.410)	1 (0.448)	7 (5.820)
	30'	1 (0.354)	**0** (0.213)	1 (0.300)	6 (0.741)	**0** (0.296)	**0** (0.348)	1 (0.325)	7 (4.540)
$100 \times 20 \times 01$	3'	2 (1.754)	**0** (1.335)	2 (1.720)	6 (5.637)	2 (1.703)	**0** (0.824)	4 (1.928)	7 (8.902)
	15'	**0** (0.824)	**0** (0.734)	4 (1.062)	6 (2.332)	**0** (0.692)	**0** (0.471)	4 (1.144)	7 (7.426)
	30'	**0** (0.544)	**0** (0.513)	4 (0.759)	6 (1.483)	**0** (0.485)	**0** (0.371)	4 (0.845)	7 (6.354)
FSP-3									
$020 \times 05 \times 01$	1'	5 (4.245)	2 (2.397)	**0** (1.057)	2 (1.831)	5 (4.837)	1 (1.891)	**0** (0.978)	6 (6.012)
	5'	4 (1.460)	4 (1.413)	**0** (0.185)	2 (0.397)	4 (1.457)	2 (0.319)	**0** (0.156)	4 (1.144)
	10'	5 (8.216)	5 (0.950)	**0** (0.103)	2 (0.206)	5 (0.903)	2 (0.160)	**0** (0.078)	3 (0.478)
$020 \times 10 \times 01$	1'	4 (1.395)	3 (1.168)	**0** (0.457)	2 (0.825)	6 (0.210)	2 (1.040)	**0** (0.533)	5 (2.936)
	5'	3 (0.493)	4 (0.619)	**0** (0.137)	2 (0.277)	7 (0.808)	3 (0.482)	**0** (0.098)	3 (0.406)
	10'	5 (0.326)	6 (0.434)	**0** (0.051)	2 (0.152)	6 (0.514)	4 (0.265)	**0** (0.046)	2 (0.180)
$020 \times 20 \times 01$	1'	2 (1.714)	**0** (1.451)	**0** (1.041)	2 (1.682)	6 (2.066)	2 (1.626)	**0** (1.118)	7 (4.945)
	5'	3 (0.622)	5 (0.890)	**0** (0.178)	2 (0.319)	5 (0.907)	3 (0.952)	**0** (0.180)	3 (0.620)
	10'	5 (0.375)	5 (0.522)	**0** (0.097)	2 (0.159)	6 (0.600)	4 (0.464)	**0** (0.069)	3 (0.320)
$050 \times 05 \times 01$	2'	4 (0.864)	**0** (0.428)	2 (0.725)	6 (1.640)	2 (0.673)	**0** (0.400)	3 (0.815)	7 (8.373)
	10'	3 (0.377)	**0** (0.332)	**0** (0.266)	1 (0.469)	6 (0.339)	3 (0.291)	**0** (0.292)	7 (6.719)
	20'	2 (0.308)	2 (0.314)	**0** (0.175)	2 (0.284)	2 (0.297)	2 (0.283)	1 (0.226)	7 (5.412)
$050 \times 10 \times 01$	2'	3 (1.546)	**0** (1.087)	4 (1.836)	6 (2.780)	2 (1.358)	**0** (1.058)	5 (2.495)	7 (8.698)
	10'	2 (0.662)	**0** (0.499)	4 (0.856)	5 (1.439)	**0** (0.558)	**0** (0.477)	5 (1.406)	7 (7.400)
	20'	2 (0.436)	**0** (0.335)	1 (0.454)	5 (0.947)	**0** (0.368)	**0** (0.320)	5 (0.962)	7 (6.522)
$050 \times 20 \times 01$	2'	3 (1.761)	1 (1.509)	4 (2.489)	5 (3.385)	1 (1.582)	**0** (1.330)	5 (3.392)	7 (8.961)
	10'	**0** (0.751)	2 (0.808)	4 (1.175)	5 (1.626)	**0** (0.664)	**0** (0.672)	6 (2.063)	7 (7.889)
	20'	**0** (0.500)	3 (0.605)	1 (0.602)	5 (1.106)	**0** (0.412)	**0** (0.451)	6 (1.470)	7 (7.094)
$100 \times 10 \times 01$	3'	3 (3.115)	**0** (2.102)	**0** (2.171)	6 (6.450)	3 (2.872)	**0** (2.012)	5 (3.559)	7 (9.466)
	15'	3 (1.192)	**0** (0.554)	2 (1.038)	6 (3.874)	3 (1.213)	**0** (0.577)	5 (1.728)	7 (9.240)
	30'	2 (0.716)	**0** (0.228)	2 (0.772)	6 (2.981)	2 (0.700)	**0** (0.235)	5 (1.186)	7 (9.084)
$100 \times 20 \times 01$	3'	2 (4.036)	**0** (3.354)	2 (3.908)	6 (8.127)	2 (3.843)	1 (3.593)	5 (5.868)	7 (9.743)
	15'	2 (1.797)	**0** (1.063)	4 (2.117)	6 (5.193)	2 (1.676)	1 (1.370)	5 (3.919)	7 (9.540)
	30'	2 (1.084)	**0** (0.412)	4 (1.590)	6 (3.888)	2 (0.973)	1 (0.594)	5 (3.054)	7 (9.441)

TABLE 2.5 – Comparaison des algorithmes DMLS selon l'indicateur $I^1_{\epsilon+}$. La première valeur représente le nombre d'algorithmes qui domine significativement l'algorithme considéré. La valeur entre parenthèses correspond à la $I^1_{\epsilon+}$-valeur moyenne normalisée ($\times 10^{-1}$).

Instance	Temps d'exécution	DMLS $(1 \cdot 1)$	DMLS $(1 \cdot 1_{\not\prec})$	DMLS $(1 \cdot 1_{\succ})$	DMLS $(1 \cdot \star)$	DMLS $(\star \cdot 1)$	DMLS $(\star \cdot 1_{\not\prec})$	DMLS $(\star \cdot 1_{\succ})$	DMLS $(\star \cdot \star)$
FSP-2									
$020 \times 05 \times 01$	1'	4 (6.284)	5 (8.256)	**0** (1.378)	**0** (1.278)	4 (7.315)	4 (6.512)	**0** (1.020)	3 (2.612)
	5'	4 (1.809)	6 (5.349)	**0** (0.518)	2 (0.745)	6 (4.325)	4 (2.177)	**0** (0.593)	2 (0.736)
	10'	4 (1.210)	6 (3.058)	**0** (0.338)	2 (0.590)	6 (2.946)	4 (1.527)	**0** (0.421)	2 (0.662)
$020 \times 10 \times 01$	1'	4 (4.989)	4 (5.374)	**0** (3.018)	**0** (3.014)	7 (7.068)	4 (5.076)	**0** (2.576)	**0** (3.424)
	5'	4 (2.942)	4 (3.038)	**0** (1.159)	2 (1.820)	7 (4.715)	4 (3.040)	**0** (1.085)	**0** (1.638)
	10'	3 (1.943)	4 (2.558)	**0** (0.500)	2 (1.250)	7 (3.626)	4 (2.302)	**0** (5.042)	2 (1.054)
$020 \times 20 \times 01$	1'	2 (3.186)	3 (3.243)	**0** (1.090)	2 (2.421)	7 (4.877)	2 (3.007)	**0** (1.382)	2 (2.846)
	5'	3 (1.034)	5 (1.625)	**0** (0.397)	**0** (0.512)	7 (2.902)	3 (1.206)	**0** (0.480)	3 (0.860)
	10'	4 (0.585)	5 (1.067)	**0** (0.224)	**0** (0.316)	7 (2.319)	4 (0.771)	**0** (0.246)	2 (0.382)
$050 \times 05 \times 01$	2'	2 (0.664)	2 (0.634)	**0** (0.356)	2 (0.553)	6 (0.809)	2 (0.577)	**0** (0.383)	7 (0.137)
	10'	3 (0.430)	3 (0.418)	**0** (0.318)	**0** (0.315)	7 (0.601)	3 (0.406)	**0** (0.297)	3 (0.371)
	20'	3 (0.376)	3 (0.371)	**0** (0.264)	**0** (0.269)	7 (0.539)	3 (0.369)	**0** (0.289)	2 (0.339)
$050 \times 10 \times 01$	2'	1 (1.045)	**0** (1.018)	**0** (0.936)	3 (1.349)	3 (1.206)	**0** (1.091)	**0** (0.937)	7 (4.483)
	10'	2 (0.689)	4 (0.934)	**0** (0.417)	2 (0.658)	4 (0.812)	6 (1.031)	**0** (0.433)	2 (0.778)
	20'	3 (0.538)	6 (0.902)	**0** (0.326)	**0** (0.393)	5 (0.672)	6 (0.903)	**0** (0.334)	3 (0.483)
$050 \times 20 \times 01$	2'	**0** (0.845)	**0** (0.828)	**0** (0.963)	5 (1.440)	1 (0.950)	**0** (0.854)	5 (1.233)	7 (5.280)
	10'	**0** (0.480)	5 (0.700)	**0** (0.447)	**0** (0.484)	3 (0.560)	3 (0.656)	**0** (0.527)	7 (1.306)
	20'	2 (0.393)	5 (0.691)	**0** (0.262)	**0** (0.328)	4 (0.500)	5 (0.653)	**0** (0.253)	4 (0.568)
$100 \times 10 \times 01$	3'	4 (0.714)	**0** (0.424)	1 (0.507)	6 (3.319)	5 (0.829)	**0** (0.484)	3 (0.626)	7 (7.143)
	15'	1 (0.339)	**0** (0.237)	**0** (0.288)	6 (1.078)	1 (0.347)	**0** (0.310)	2 (0.340)	7 (5.273)
	30'	**0** (0.261)	**0** (0.195)	**0** (0.210)	6 (0.583)	**0** (0.241)	**0** (0.280)	**0** (0.243)	7 (4.207)
$100 \times 20 \times 01$	3'	2 (1.106)	**0** (0.817)	2 (1.232)	6 (4.814)	2 (1.076)	**0** (0.824)	4 (1.295)	7 (8.057)
	15'	**0** (0.501)	**0** (0.471)	4 (0.742)	6 (1.811)	**0** (0.406)	**0** (0.471)	4 (0.729)	7 (6.314)
	30'	**0** (0.322)	**0** (0.309)	4 (0.526)	6 (1.131)	**0** (0.271)	**0** (0.371)	4 (0.508)	7 (5.263)
FSP-3									
$020 \times 05 \times 01$	1'	5 (9.072)	2 (7.088)	**0** (3.880)	1 (5.902)	5 (9.064)	1 (5.643)	**0** (4.550)	5 (9.302)
	5'	4 (6.121)	4 (5.390)	**0** (1.781)	2 (3.121)	5 (6.756)	2 (2.188)	**0** (1.784)	3 (4.493)
	10'	5 (4.789)	5 (4.540)	**0** (1.572)	2 (2.162)	5 (4.977)	**0** (1.645)	**0** (1.472)	1 (2.558)
$020 \times 10 \times 01$	1'	3 (2.715)	3 (2.508)	**0** (1.829)	2 (2.225)	6 (3.370)	2 (1.879)	**0** (1.991)	5 (4.087)
	5'	3 (1.813)	4 (2.023)	**0** (0.903)	2 (1.440)	5 (2.074)	4 (1.950)	**0** (0.710)	2 (1.603)
	10'	4 (1.754)	4 (1.847)	**0** (0.578)	2 (1.290)	5 (1.879)	3 (1.545)	**0** (4.615)	2 (1.206)
$020 \times 20 \times 01$	1'	1 (2.545)	**0** (2.834)	**0** (1.940)	**0** (2.806)	1 (2.680)	2 (3.002)	**0** (2.176)	7 (6.189)
	5'	3 (1.069)	4 (1.907)	**0** (0.290)	2 (0.906)	4 (1.337)	4 (1.939)	1 (0.426)	3 (1.435)
	10'	3 (0.711)	5 (1.236)	**0** (0.116)	2 (0.384)	5 (1.010)	4 (0.986)	**0** (0.151)	3 (0.861)
$050 \times 05 \times 01$	2'	4 (0.489)	**0** (0.289)	2 (0.387)	6 (1.064)	2 (0.351)	**0** (0.309)	3 (0.463)	7 (7.082)
	10'	**0** (0.304)	**0** (0.282)	**0** (0.295)	**0** (0.314)	**0** (0.304)	**0** (0.304)	**0** (0.304)	7 (5.433)
	20'	**0** (0.304)	**0** (0.282)	**0** (0.271)	**0** (0.297)	**0** (0.304)	**0** (0.304)	**0** (0.304)	7 (4.279)
$050 \times 10 \times 01$	2'	2 (0.884)	**0** (0.625)	4 (1.054)	5 (1.586)	2 (0.813)	**0** (0.617)	5 (1.511)	7 (7.274)
	10'	**0** (0.395)	**0** (0.377)	1 (0.479)	5 (0.798)	**0** (0.416)	**0** (0.382)	5 (0.859)	7 (5.689)
	20'	**0** (0.280)	**0** (0.323)	**0** (0.280)	5 (0.537)	**0** (0.312)	**0** (0.331)	5 (0.580)	7 (0.482)
$050 \times 20 \times 01$	2'	2 (0.870)	**0** (0.691)	4 (1.285)	5 (1.875)	1 (0.764)	**0** (0.642)	5 (1.960)	7 (7.441)
	10'	**0** (0.357)	**0** (0.435)	2 (0.482)	5 (0.764)	**0** (0.348)	**0** (0.351)	6 (1.058)	7 (5.864)
	20'	**0** (0.255)	2 (0.348)	**0** (0.235)	5 (0.485)	**0** (0.229)	**0** (0.252)	6 (0.701)	7 (4.995)
$100 \times 10 \times 01$	3'	3 (1.546)	**0** (0.998)	2 (1.238)	6 (4.220)	3 (1.452)	**0** (0.959)	5 (2.070)	7 (7.818)
	15'	2 (0.553)	**0** (0.264)	2 (0.570)	6 (2.320)	2 (0.576)	**0** (0.256)	5 (1.011)	7 (7.370)
	30'	2 (0.329)	**0** (0.137)	4 (0.422)	6 (1.782)	2 (0.335)	**0** (0.115)	5 (0.694)	7 (7.083)
$100 \times 20 \times 01$	3'	1 (2.125)	**0** (1.648)	3 (2.424)	6 (6.119)	1 (1.907)	1 (1.935)	5 (3.920)	7 (8.665)
	15'	2 (0.809)	**0** (0.381)	4 (1.216)	6 (3.490)	2 (0.742)	1 (0.582)	5 (2.492)	7 (8.131)
	30'	2 (0.477)	**0** (0.099)	4 (0.943)	6 (2.450)	2 (0.409)	1 (0.219)	5 (1.906)	7 (7.943)

est la meilleure stratégie à adopter pour cet ensemble d'instances, ceci quelle que soit la méthode de sélection de l'ensemble courant.
Pour les instances de grande taille, la méthode d'exploration du voisinage la plus efficace correspond clairement à la stratégie « 1er voisin non-dominé ». En effet, DMLS $(1 \cdot 1_{\not\prec})$ et DMLS $(\star \cdot 1_{\not\prec})$ ne sont jamais tous les deux surpassés par une même méthode. Toutefois, même s'il est souvent devancé par les méthodes mentionnées ci-dessus, DMLS $(1 \cdot 1_{\succ})$ obtient également de bonnes performances moyennes sur des instances de grande taille, contrairement à DMLS $(\star \cdot 1_{\succ})$ qui présente de mauvaises performances, en particulier pour les instances du problème FSP-3. En outre, les résultats obtenus par DMLS $(\star \cdot \star)$ sont très faibles, corroborant les résultats mis en évidence pour la première série d'instances. À l'inverse, les méthodes relativement simples comme DMLS $(1 \cdot 1)$ et DMLS $(\star \cdot 1)$ obtiennent de bonnes performances moyennes sur cet ensemble d'instances de grande taille.

Pour résumer, il n'existe pas de stratégie claire à adopter pour la sélection de l'ensemble courant. Toutefois, l'arrêt de l'exploration du voisinage d'une solution une fois le premier voisin dominant trouvé semble être très efficace pour les instances de petite taille. Pour les problèmes de plus grande taille, stopper l'exploration du voisinage lors de l'évaluation de la première solution voisine non-dominée s'avère plus performant. Il se peut qu'un tel comportement soit lié au nombre de redémarrages effectués par les algorithmes. En effet, comme indiqué dans le tableau 2.6, DMLS $(1 \cdot 1_{\succ})$ et DMLS $(\star \cdot 1_{\succ})$ redémarrent tous deux plus souvent que DMLS $(1 \cdot 1_{\not\prec})$ et DMLS $(\star \cdot 1_{\not\prec})$ pour les petites instances, où ils obtiennent de meilleurs résultats. Mais pour les plus grandes instances, où on obtient le cas inverse, le nombre de redémarrages est soit proche de zéro, soit plus ou moins comparable entre les deux classes de méthode. Notez que ceci pourrait également être lié au nombre de points situés sur le front Pareto pour l'instance considérée. Comme indiqué dans le même tableau, ce nombre est plus élevé pour les instances de grande taille. Or, nous pouvons raisonnablement supposer que, plus le nombre de solutions non-dominées est élevé, plus grand sera le nombre de solutions non-dominées effectivement trouvées par les algorithmes, et donc plus grande sera la taille de l'archive. En conséquence, la chance de tomber dans une sorte d'optimum local Pareto est réduite, tout comme la chance de redémarrer le processus de recherche.

2.3.1.5 Discussion

En termes purement conceptuel, DMLS $(\star \cdot \star)$ semble à première vue être la méthode la plus appropriée si l'on considère qu'un de temps de calcul illimité est disponible. En effet, sa capacité d'exploration de l'espace de recherche est probablement la plus élevée par rapport aux autres approches DMLS. Toutefois, en pratique, cet algorithme semble rapidement limité par les ressources vis-à-vis de la taille du front de Pareto. L'algorithme DMLS $(1 \cdot \star)$ surmonte ces limites en réduisant la complexité en temps et en espace d'une itération. En effet, il évalue un sous-ensemble de voisins de taille polynomiale à chaque itération, et le nombre d'évaluations exhaustives du voisinage est donc réduit. Dans la pratique, il obtient globalement de meilleurs résultats que DMLS $(\star \cdot \star)$, au moins pour les instances de grande taille. D'autre part, et en opposition aux méthodes mentionnées ci-dessus, DMLS $(1 \cdot 1)$ et DMLS $(\star \cdot 1)$ apparaissent de prime abord un peu plus aléatoires. Ils offrent de bonnes possibilités en termes de diversification, mais présentent un manque d'intensification, la pression d'intensification étant appliquée uniquement au cours de l'étape d'archivage. En pratique, nos résultats indiquent leur performance décente pour les

TABLE 2.6 – Nombre moyen de redémarrages aléatoires au cours de la recherche. Pour information, la taille du meilleur ensemble non-dominé trouvé est donnée dans la deuxième colonne (taille de l'ens. de référence).

Instance	taille de l'ens. de référence	DMLS $(1 \cdot 1)$	DMLS $(1 \cdot 1_{\not\prec})$	DMLS $(1 \cdot 1_{\succ})$	DMLS $(1 \cdot \star)$	DMLS $(\star \cdot 1)$	DMLS $(\star \cdot 1_{\not\prec})$	DMLS $(\star \cdot 1_{\succ})$	DMLS $(\star \cdot \star)$
FSP-2									
$020 \times 05 \times 01$	18	977.80	86.15	2878.20	1100.05	4932.15	206.35	2228.25	430.70
$020 \times 10 \times 01$	24	458.85	47.10	392.10	226.25	1789.10	68.55	265.05	97.75
$020 \times 20 \times 01$	30	329.30	18.85	220.10	151.95	1239.65	32.30	145.70	66.85
$050 \times 05 \times 01$	28	85.20	4.70	77.40	22.05	141.15	4.50	56.40	6.40
$050 \times 10 \times 01$	48	37.10	1.20	14.45	7.35	63.75	0.30	8.85	1.60
$050 \times 20 \times 01$	38	18.70	1.00	6.40	4.00	46.75	0.00	3.45	0.70
$100 \times 10 \times 01$	89	1.00	0.00	0.40	0.00	0.00	0.00	0.00	0.00
$100 \times 20 \times 01$	76	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
FSP-3									
$020 \times 05 \times 01$	278	19.85	11.65	52.00	17.40	273.15	35.20	34.60	6.60
$020 \times 10 \times 01$	191	38.40	10.65	75.00	35.05	406.80	23.15	42.80	10.95
$020 \times 20 \times 01$	423	11.85	3.10	18.40	9.80	133.10	2.00	8.75	3.15
$050 \times 05 \times 01$	273	1.65	1.00	3.70	0.70	0.00	0.00	1.70	0.00
$050 \times 10 \times 01$	969	1.10	1.00	0.00	0.00	0.00	0.00	0.00	0.00
$050 \times 20 \times 01$	592	1.25	1.00	0.00	0.00	0.00	0.00	0.00	0.00
$100 \times 10 \times 01$	696	1.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00
$100 \times 20 \times 01$	618	1.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00

instances où un grand nombre de solutions non-dominées sont susceptibles d'être trouvées.
En conséquence, tout en gardant à l'esprit de trouver un équilibre idéal entre intensification et diversification, guider l'exploration du voisinage par une stratégie fondée sur une relation de dominance semble être plus fiable. Tout d'abord, en comparaison à l'exploration d'un seul voisin aléatoire, les algorithmes DMLS $(1 \cdot 1_{\not\prec})$ et DMLS $(\star \cdot 1_{\not\prec})$ offrent une aptitude de diversification équivalente. Mais ces derniers ne consomment pas de ressource de calcul pour traiter les solutions dominées (et donc peu attrayantes). En outre, restreindre la probabilité d'accepter une solution voisine augmente les chances d'obtenir une solution améliorante vis-à-vis de l'approximation courante. Il existe donc une amélioration indirecte du degré d'intensification. Les tendances générales des résultats sont assez similaires à ceux reportés pour DMLS $(1 \cdot 1)$ et DMLS $(\star \cdot 1)$, mais avec des performances largement plus élevées.
Enfin, la stratégie du « 1er voisin dominant » semble être le compromis idéal entre la stratégie du « 1er voisin non-dominé » et la stratégie exhaustive, et fournit donc un intéressant compromis exploration-exploitation. En effet, les capacités d'intensification sont nettement améliorées par rapport à DMLS $(1 \cdot 1_{\not\prec})$ et DMLS $(\star \cdot 1_{\not\prec})$, tout en conservant une diversification raisonnable, étant donné que les voisins non-dominés trouvés au cours de l'exploration du voisinage sont toujours proposés pour l'archivage. En outre, en comparaison à DMLS $(1 \cdot \star)$ et DMLS $(\star \cdot \star)$, la convergence vers un ensemble localement optimal est accélérée, bien que ces derniers soient plus à même de se diriger de façon plus proche vers le front Pareto à une itération donnée. Dans la pratique, DMLS $(1 \cdot 1_{\succ})$ et DMLS $(\star \cdot 1_{\succ})$ sont globalement les méthodes plus performantes pour presque la totalité de nos expériences, à l'exception des instances de grande taille où les méthodes pour lesquelles la diversité est légèrement plus encouragée semblent plus performantes.

Maintenant, en ce qui concerne la sélection de l'ensemble courant, l'analyse expérimentale ne permet pas de fournir une conclusion claire sur la stratégie à adopter. Toutefois, il semble que la sélection d'une seule solution aléatoire soit légèrement plus efficace, car les approches correspondantes obtiennent plus souvent de meilleurs résultats que leurs homologues. Toutefois, puisque

nœud	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}
clé	0	0.7	-	0.3	-	0.8	0.2	-	0.5	-

Fig. 2.20 – Une solution au problème de Ring-Star représentée à l'aide de clés aléatoires.

sélectionner une solution unique permet d'éviter d'explorer le voisinage de certaines solutions (les voisins trouvés étant susceptibles d'engendrer la suppression de solutions de l'archive), cette question est en quelque sorte également liée à l'équilibre entre intensification et diversification.

2.3.2 Application au problème de Ring-Star

Cette section se concentre sur la recherche d'une approximation de l'ensemble Pareto optimal pour le problème de Ring-Star introduit à la section 1.4.2. Pour cela, nous allons évaluer expérimentalement les performances d'un ensemble de métaheuristiques, à la fois des algorithmes évolutionnaires et des algorithmes de recherche locale. Quatre méthodes sont ici développées pour le problème traité. Ainsi, une recherche locale à base de population, deux algorithmes évolutionnaires classiques et un troisième expérimenté ici pour la première fois sont comparés les uns aux autres pour la résolution d'instances impliquant jusqu'à 300 nœuds.
Tout d'abord, une modélisation du problème destinée à une résolution par métaheuristique est proposée. Celle-ci peut également être vue comme une illustration d'application des différents modèles de conception proposés. Ensuite, sont dévoilées les méthodologies choisies pour une telle résolution. Elles sont suivies de la présentation et d'une discussion des résultats expérimentaux produits par la comparaison de ces méthodes.

2.3.2.1 Modélisation

Cette section présente les composants spécifiquement conçus et implémentés pour le problème de Ring-Star. Ces éléments sont nécessaires pour instancier les différentes métaheuristiques appliquées à la résolution du problème. Par conséquent, la représentation, l'évaluation, l'initialisation ainsi que les opérateurs de voisinage, de mutation et de croisement sont détaillés ci-dessous.

2.3.2.1.1 Représentation. L'encodage que nous allons utiliser pour représenter une solution au problème de Ring-Star est basé sur le mécanisme de clés aléatoires (*random keys*) proposé par Bean (1994). Une telle représentation a déjà été appliquée avec succès lors de la résolution d'une version monoobjectif du problème traité (Renaud *et al.*, 2004). Elle consiste à attribuer, à chaque nœud appartenant à l'anneau $v_i \in V'$, une clé aléatoire $k_i \in [0, 1[$, avec $k_1 = 0$. Une valeur particulière est affectée aux nœuds non visités. Ainsi, le chemin de l'anneau associé à une solution correspond à l'ordre des clés aléatoires des nœuds lus dans l'ordre croissant ; si $k_i < k_j$ alors v_j se situe après v_i. Une représentation possible pour le cycle (v_1,v_7,v_4,v_9,v_2,v_6) est donnée sur la figure 2.20. Les nœuds non visités v_3,v_5,v_8 et v_{10} sont affectés à un nœud visité de sorte que le coût d'affectation associé est minimum.

2.3.2.1.2 Évaluation. L'étape d'évaluation consiste à calculer les valeurs d'une solution donnée pour chacune des fonctions objectif, c'est-à-dire, pour le problème de Ring-Star, le coût de l'anneau et le coût d'affectation.

2.3.2.1.3 Initialisation. Les solutions initiales sont générées de façon aléatoire. Chaque nœud a une probabilité $p = 0.5$ d'être visité ou non, et à chaque nœud visité $v_i \in V'$ est associée une clé aléatoire k_i uniformément générée dans l'intervalle $[0, 1[$.

2.3.2.1.4 Voisinage. Le problème de Ring-Star étant à la fois un problème de tournée et un problème d'affectation, différents opérateurs de voisinage doivent être conçus. Pour ce type de problème, les opérateurs habituels consistent à supprimer ou à ajouter un nœud à l'anneau. Ici, nous considérons également un opérateur spécialement consacré à l'amélioration du cycle par le biais d'un échange de type 2-opt. Les trois opérateurs sont donc les suivants.

- **Suppression**. Cet opérateur consiste à sélectionner un nœud visité $v_i \in V'$ et à le supprimer de l'anneau. Ce nœud devient donc non-visité.
- **Insertion**. Cet opérateur sélectionne un nœud non visité $v_i \in V \setminus V'$ et l'ajoute au cycle. La position d'insertion de v_i est choisie de sorte que la valeur incrémentale du coût de l'anneau soit minimum.
- **Échange 2-opt**. Deux nœuds visités v_i et $v_j \in V'$ sont sélectionnés, et un mouvement de type 2-opt est appliqué entre v_i et v_j : la séquence de nœuds de l'anneau situés entre v_i et v_j est inversée.

Notez que les voisins d'une solution donnée sont explorés de façon unique et aléatoire, sans considérer aucun ordre entre les trois opérateurs.

2.3.2.1.5 Évaluation incrémentale. Après l'application d'un mouvement, il n'est pas nécessaire de réévaluer une solution complètement. Ainsi, après avoir un mouvement de type suppression, outre la différence engendrée sur le coût de l'anneau, il suffit de réassigner les nœuds précédemment attribués à celui qui a été supprimé. De même, après avoir un mouvement de type insertion, il faut réaffecter les nœuds non-visités de sorte que le coût d'affectation soit minimum. Enfin, après un mouvement de type échange 2-opt, il suffit de calculer la différence engendrée sur le coût de l'anneau, le coût d'affectation restant inchangé.

2.3.2.1.6 Variation. Ci-dessous sont décrits les opérateurs de variation développés pour l'application d'algorithmes évolutionnaires au problème de Ring-Star.

Croisement. L'opérateur de croisement est un croisement 1-point étroitement lié à celui proposé par Renaud *et al.* (2004). Deux solutions x_1 et x_2 sont divisées à une position aléatoire. La partie gauche de x_1 est alors associée à la partie droite de x_2 pour construire un premier individu Enfant, et la partie gauche de x_2 est combinée à la partie droite de x_1 pour construire un deuxième individu Enfant. Chaque nœud conserve sa clé aléatoire de sorte qu'une simple reconstruction des nouveaux individus est permise. Grâce au mécanisme d'encodage basé sur les clés aléatoires, des solutions dont les anneaux sont de tailles différentes peuvent très facilement être recombinées, même si les structures initiales des anneaux sont en général légèrement brisées chez les solutions Enfant. La figure 2.21 illustre un croisement entre deux solutions (v_1,v_7,v_4,v_9,v_2,v_6) et (v_1,v_8,v_4,v_3,v_5) au niveau du nœud v_6, donnant naissance à deux nouvelles solutions (v_1,v_8,v_4,v_2,v_6) et (v_1,v_7,v_9,v_4,v_3,v_5).

Notez qu'un autre opérateur préservant une plus grande part des structures initiales des anneaux a été réalisé. Mais celui-ci avait tendance à réduire le nombre de nœuds appartenant au cycle, ce

parent 1

nœud	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}
clé	0	0.7	-	0.3	-	0.8	0.2	-	0.5	-

parent 2

nœud	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}
clé	0	-	0.8	0.7	0.9	-	-	0.2	-	-

enfant 1

nœud	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}
clé	0	0.7	-	0.3	-	0.8	-	0.2	-	-

enfant 2

nœud	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}
clé	0	-	0.8	0.7	0.9	-	0.2	-	0.5	-

Fig. 2.21 – Opérateur de croisement pour le problème de Ring-Star.

qui causait une convergence prématurée des algorithmes vers des solutions ayant un petit nombre de nœuds visités.

Mutation. Trois opérateurs de mutation sont considérés. Ils sont identiques aux opérateurs de voisinage précédemment présentés, à la différence près qu'ils sont ici appliqués à des nœuds choisis aléatoirement.

2.3.2.2 Approches étudiées

L'analyse expérimentale menée par la suite va consister à comparer différentes méthodes, à la fois classiques et novatrices, pour la résolution du problème de Ring-Star. Tout d'abord, IBMOLS (Basseur et Burke, 2007), recherche locale dont la stratégie d'affectation des valeurs de fitness est fondée sur un indicateur binaire de qualité, est le premier algorithme considéré. Ensuite, deux algorithmes évolutionnaires couramment utilisés en optimisation multiobjectif sont appliqués au problème traité. Il s'agit des algorithmes NSGA-II (Deb *et al.*, 2002) et IBEA (Zitzler et Künzli, 2004) introduits dans la section 2.1.4.1. Le premier est considéré comme une référence dans le domaine de l'optimisation multiobjectif évolutionnaire, et le second est un algorithme plus moderne et est en quelque sorte l'inspirateur de la recherche locale IBMOLS également étudiée. Par rapport aux algorithmes originaux, nous ajoutons ici une archive qui est mise à jour à chaque itération de l'algorithme ; ceci afin de conserver la totalité des solutions non-dominées trouvées depuis le début de la recherche. Enfin, la dernière métaheuristique consiste en l'algorithme SEEA proposé dans la section 2.1.4.1 et appliqué ici pour la première fois. La technique d'archivage mise en place au sein de chacune des méthodes est non bornée.

2.3.2.3 Design expérimental

Tous les algorithmes ont été implémentés sous ParadisEO version 1.1, et les expérimentations ont été menées sur une machine dotée d'un processeur Intel Core 2 Duo 6600 (2×2.40 GHz, 2 Go RAM) avec G++ 4.1.2 fonctionnant sous Linux. Les algorithmes appliqués au problème de Ring-Star sont donc comparés de façon équitable.
Les conditions d'arrêt et de reprise considérées ainsi que les paramètres utilisés sont présentés ci-dessous.

TABLE 2.7 – Condition d'arrêt : temps de calcul par exécution.

Instance	Temps d'exécution	Instance	Temps d'exécution
eil51	20"	*kroA150*	10'
st70	1'	*kroA200*	20'
kroA100	2'	*pr264*	30'
bier127	5'	*pr299*	50'

2.3.2.3.1 Conditions d'arrêt et de reprise. Pour des raisons identiques à celles qui ont été exposées lors de l'analyse expérimentale menée sur le problème de Flowshop, la condition d'arrêt que nous avons considérée se base sur un temps maximum d'exécution. En effet, de notre point de vue, il n'existe pas d'approche standard pour définir une condition d'arrêt incontestable dans le cadre de métaheuristiques pour l'optimisation multiobjectif. Les stratégies habituelles sont très basiques et consistent généralement en un nombre arbitraire d'itérations ou d'évaluations. Néanmoins, dans notre cas, une itération pour un algorithme évolutionnaire n'a rien à voir avec une itération pour un algorithme de recherche locale. En outre, le calcul du nombre d'évaluations effectuées depuis le début du processus de recherche n'a pas de sens ici, car les solutions voisines et les solutions mutées ne sont pas évaluées complètement, mais de façon incrémentale. Par conséquent, dans le cadre de cette étude, nous avons décidé d'arrêter le processus de recherche après un certain temps d'exécution. En effet, toutes les méthodes partagent les mêmes composants de base et leurs implémentations respectives peuvent donc être comparées en termes de temps de calcul. Mais, les petites instances sont supposément plus faciles à résoudre que les plus grandes, ce critère d'arrêt a donc été fixé en fonction de la taille de l'instance considérée, comme indiqué dans le tableau 2.7. Notez que le temps maximum d'exécution disponible est stipulé pour une seule simulation par instance et par algorithme.

De même que pour les algorithmes de recherche locale appliqués au problème de Flowshop, permettez-nous de rappeler que l'algorithme IBMOLS possède une condition d'arrêt naturelle (archive non améliorante lors de la dernière itération). Toutefois, la répétition itérative d'un algorithme de recherche locale conduit souvent à des performances très élevées et peut améliorer considérablement les résultats. Dans le cadre de l'optimisation monoobjectif, deux stratégies principales de recherche locale itérée peuvent être distinguées : la reprise du processus à l'aide d'une nouvelle solution aléatoire (*random restart*) et les méthodes de perturbation (Lourenco *et al.*, 2002). Le redémarrage aléatoire constitue la possibilité la plus simple, mais une technique de perturbation adaptée permet généralement d'obtenir de meilleurs résultats. Néanmoins, avec IBMOLS, nous devons maintenant composer avec une population de solutions qui doit être réinitialisée entre chaque phase de recherche locale. Comme proposé par Basseur et Burke (2007) parmi d'autres alternatives, nous décidons de perturber un ensemble de solutions provenant de l'archive pour constituer la nouvelle population. Le choix d'une telle stratégie a été motivé par des expérimentations préliminaires ainsi que des résultats probants obtenus par Basseur et Burke (2007) pour une autre application. Ainsi, la stratégie de réinitialisation de la population se base sur un bruit aléatoire, tout comme dans un algorithme de recuit simulé de base (Burke et Kendall, 2005). Ce bruit se compose de multiples mutations appliquées en série à un ensemble de N solutions provenant de l'archive. Notez que si la taille de l'archive est inférieure à N, la population est complétée à l'aide de solutions aléatoires, tout comme lors d'une stratégie de reprise aléatoire. Pour éviter les confusions, la version itérée de l'algorithme IBMOLS sera notée

Table 2.8 – Paramètres.

	I-IBMOLS		IBEA	NSGA-II	SEEA
Instance	taille de la pop.	taux de bruit	taille de la pop.	taille de la pop.	taille de la pop.
eil51	20	10%	100	100	100
st70	20	10%	50	100	100
kroA100	30	10%	100	200	100
bier127	30	10%	100	200	100
kroA150	30	10%	200	200	100
kroA200	30	10%	200	200	100
pr264	30	20%	50	200	100
pr299	50	10%	50	200	100

I-IBMOLS (*iterated IBMOLS*).

2.3.2.3.2 Paramètres. Une phase expérimentale préliminaire a été effectuée afin de déterminer bon nombre des paramètres suivants (Liefooghe *et al.*, 2008e). Néanmoins, pour certains algorithmes, aucune tendance générale n'a été identifiée quant à certains paramètres, la taille de la population a donc été fixée en fonction de l'instance considérée, comme indiqué dans le tableau 2.8. Le taux de bruit de l'algorithme I-IBMOLS a été fixé à un pourcentage du nombre de nœuds pour le problème traité. Ensuite, tout comme initialement proposé par Zitzler et Künzli (2004), le facteur d'échelle κ a été fixé à 0.05 pour les métaheuristiques à base d'indicateur (IBEA et I-IBMOLS). Les autres paramètres sont partagés par tous les algorithmes évolutionnaires et se composent d'une probabilité de croisement de 0.25, d'une probabilité de mutation de 1.00, avec des taux de 0.25, 0.25 et 0.50 pour les opérateurs de mutation de type suppression, insertion et échange 2-opt, respectivement.

2.3.2.4 Résultats expérimentaux

Les tableaux 2.9 et 2.10 fournissent une comparaison des algorithmes SEEA, NSGA-II, IBEA et I-IBMOLS par rapport à l'indicateur I_H^- et à l'indicateur $\mathrm{I}_{\epsilon+}^1$, respectivement.
Les résultats obtenus donnent un avantage indéniable à I-IBMOLS. Cet algorithme n'est en effet jamais statistiquement surpassé par aucun autre algorithme selon les deux indicateurs de qualité. Les seules exceptions sont pour les instances *st70* et *pr264*, pour lesquelles SEEA a obtenu de meilleurs résultats par rapport à la métrique epsilon. Quant à NSGA-II, il est distancé par tous les autres algorithmes sur chacune des instances testées, sauf pour le problème *pr264* où il n'y a pas de différence significative entre les résultats de NSGA-II et ceux d'IBEA selon l'indicateur $\mathrm{I}_{\epsilon+}^1$. Enfin, par rapport à l'indicateur I_H^-, les résultats obtenus par IBEA et SEEA semblent assez hétérogènes d'une instance à une autre, de sorte qu'aucune tendance générale ne peut être identifiée. Toutefois, selon l'indicateur $\mathrm{I}_{\epsilon+}^1$, SEEA semble nettement plus efficace que l'algorithme IBEA sur la plupart des instances de test.

Comparaison aux résultats obtenus pour le problème de Ring-Star monoobjectif. Pour finir, nous fournissons une comparaison entre les résultats que nous avons obtenus pour le problème de Ring-Star biobjectif étudié ici, et ceux du problème de Ring-Star monoobjectif

TABLE 2.9 – Comparaison des algorithmes selon l'indicateur I_H^-. Pour chaque instance, soit l'algorithme situé sur une ligne donnée domine significativement l'algorithme situé sur une colonne donnée ($\succ$), soit il est significativement dominé ($\prec$), soit il n'existe pas de différence significative entre les deux algorithmes ($\equiv$). La valeur entre parenthèses correspond à la I_H^--valeur moyenne ($\times 10^{-3}$).

			I-IBMOLS	IBEA	NSGA-II	SEEA
eil51	I-IBMOLS	(4.456)	-	$\succ$	$\succ$	$\equiv$
	IBEA	(6.710)	$\prec$	-	$\succ$	$\prec$
	NSGA-II	(12.573)	$\prec$	$\prec$	-	$\prec$
	SEEA	(4.957)	$\equiv$	$\succ$	$\succ$	-
st70	I-IBMOLS	(3.143)	-	$\succ$	$\succ$	$\succ$
	IBEA	(4.037)	$\prec$	-	$\succ$	$\equiv$
	NSGA-II	(8.920)	$\prec$	$\prec$	-	$\prec$
	SEEA	(3.718)	$\prec$	$\equiv$	$\succ$	-
kroA100	I-IBMOLS	(4.251)	-	$\succ$	$\succ$	$\succ$
	IBEA	(5.273)	$\prec$	-	$\succ$	$\equiv$
	NSGA-II	(12.370)	$\prec$	$\prec$	-	$\prec$
	SEEA	(5.015)	$\prec$	$\equiv$	$\succ$	-
bier127	I-IBMOLS	(3.219)	-	$\succ$	$\succ$	$\succ$
	IBEA	(4.236)	$\prec$	-	$\succ$	$\succ$
	NSGA-II	(9.606)	$\prec$	$\prec$	-	$\prec$
	SEEA	(6.751)	$\prec$	$\prec$	$\succ$	-
kroA150	I-IBMOLS	(3.959)	-	$\succ$	$\succ$	$\succ$
	IBEA	(4.562)	$\prec$	-	$\succ$	$\equiv$
	NSGA-II	(8.973)	$\prec$	$\prec$	-	$\prec$
	SEEA	(4.747)	$\prec$	$\equiv$	$\succ$	-
kroA200	I-IBMOLS	(2.875)	-	$\equiv$	$\succ$	$\succ$
	IBEA	(2.980)	$\equiv$	-	$\succ$	$\succ$
	NSGA-II	(8.515)	$\prec$	$\prec$	-	$\prec$
	SEEA	(3.822)	$\prec$	$\prec$	$\succ$	-
pr264	I-IBMOLS	(1.535)	-	$\succ$	$\succ$	$\equiv$
	IBEA	(1.912)	$\prec$	-	$\succ$	$\prec$
	NSGA-II	(3.663)	$\prec$	$\prec$	-	$\prec$
	SEEA	(1.520)	$\equiv$	$\succ$	$\succ$	-
pr299	I-IBMOLS	(1.303)	-	$\succ$	$\succ$	$\succ$
	IBEA	(1.964)	$\prec$	-	$\succ$	$\equiv$
	NSGA-II	(4.185)	$\prec$	$\prec$	-	$\prec$
	SEEA	(2.179)	$\prec$	$\equiv$	$\succ$	-

TABLE 2.10 – Comparaison des algorithmes selon l'indicateur $I^1_{\epsilon+}$. Pour chaque instance, soit l'algorithme situé sur une ligne donnée domine significativement l'algorithme situé sur une colonne donnée ($\succ$), soit il est significativement dominé ($\prec$), soit il n'existe pas de différence significative entre les deux algorithmes ($\equiv$). La valeur entre parenthèses correspond à la $I^1_{\epsilon+}$-valeur moyenne ($\times 10^{-3}$).

			I-IBMOLS	IBEA	NSGA-II	SEEA
eil51	I-IBMOLS	(9.307)	-	$\succ$	$\succ$	$\equiv$
	IBEA	(12.094)	$\prec$	-	$\succ$	$\prec$
	NSGA-II	(19.165)	$\prec$	$\prec$	-	$\prec$
	SEEA	(9.613)	$\equiv$	$\succ$	$\succ$	-
st70	I-IBMOLS	(7.321)	-	$\succ$	$\succ$	$\prec$
	IBEA	(10.334)	$\prec$	-	$\succ$	$\prec$
	NSGA-II	(13.639)	$\prec$	$\prec$	-	$\prec$
	SEEA	(6.298)	$\succ$	$\succ$	$\succ$	-
kroA100	I-IBMOLS	(9.833)	-	$\succ$	$\succ$	$\equiv$
	IBEA	(11.771)	$\prec$	-	$\succ$	$\prec$
	NSGA-II	(17.718)	$\prec$	$\prec$	-	$\prec$
	SEEA	(9.606)	$\equiv$	$\succ$	$\succ$	-
bier127	I-IBMOLS	(8.421)	-	$\succ$	$\succ$	$\succ$
	IBEA	(11.993)	$\prec$	-	$\succ$	$\succ$
	NSGA-II	(21.522)	$\prec$	$\prec$	-	$\prec$
	SEEA	(19.377)	$\prec$	$\prec$	$\succ$	-
kroA150	I-IBMOLS	(7.853)	-	$\succ$	$\succ$	$\succ$
	IBEA	(10.708)	$\prec$	-	$\succ$	$\prec$
	NSGA-II	(13.383)	$\prec$	$\prec$	-	$\prec$
	SEEA	(9.056)	$\prec$	$\succ$	$\succ$	-
kroA200	I-IBMOLS	(7.829)	-	$\equiv$	$\succ$	$\equiv$
	IBEA	(7.288)	$\equiv$	-	$\succ$	$\succ$
	NSGA-II	(14.473)	$\prec$	$\prec$	-	$\prec$
	SEEA	(8.204)	$\equiv$	$\prec$	$\succ$	-
pr264	I-IBMOLS	(5.259)	-	$\succ$	$\succ$	$\prec$
	IBEA	(9.055)	$\prec$	-	$\equiv$	$\prec$
	NSGA-II	(8.403)	$\prec$	$\equiv$	-	$\prec$
	SEEA	(4.343)	$\succ$	$\succ$	$\succ$	-
pr299	I-IBMOLS	(4.023)	-	$\succ$	$\succ$	$\succ$
	IBEA	(8.993)	$\prec$	-	$\succ$	$\prec$
	NSGA-II	(10.403)	$\prec$	$\prec$	-	$\prec$
	SEEA	(5.768)	$\prec$	$\succ$	$\succ$	-

TABLE 2.11 – Ratio d'erreur entre la meilleure valeur trouvée au cours de nos expérimentations et le coût de la solution optimale du TSP monoobjectif ou le meilleur coût d'anneau trouvé pour le Ring-Star monobjectif par Labbé *et al.* (2004).

Instance	TSP	RSP monoobjectif			
	monoobjectif	$\alpha = 3$	$\alpha = 5$	$\alpha = 7$	$\alpha = 9$
eil51	1.10 %	1.10 %	0.75 %	0.37 %	0.69 %
st70	1.47 %	1.47 %	0.49 %	0.56 %	0.42 %
kroA100	0.64 %	0.64 %	0.10 %	0.14 %	**0.00 %**
bier127	1.16 %	1.16 %	1.97 %	0.52 %	0.05 %
kroA150	3.12 %	3.12 %	1.37 %	0.60 %	0.05 %
kroA200	2.83 %	**- 3.31 %**	3.39 %	1.29 %	**- 1.37 %**
pr264	4.60 %	-	-	-	-
pr299	3.72 %	-	-	-	-

étudié par Labbé *et al.* (2004), où les deux coûts (anneau et affectation) sont additionnés [9]. Afin de fournir des solutions optimales visitant approximativement 25, 50, 75 and 100% du nombre total de nœuds, les auteurs ont fixé le coût de l'anneau c_{ij} et le coût d'affectation d_{ij} entre deux nœuds v_i et v_j de la façon suivante : $c_{ij} = \lceil \alpha l_{ij} \rceil$ et $d_{ij} = \lceil (10 - \alpha) l_{ij} \rceil$ avec $\alpha \in \{3, 5, 7, 9\}$, où l_{ij} représente la distance entre v_i et v_j donnée par l'instance de la TSPLIB.

Afin de donner une idée générale des résultats que nous avons obtenus, nous allons comparer les meilleures valeurs scalaires trouvées telles que cela est détaillé ci-dessus à la valeur optimale trouvée par Labbé *et al.* (2004) [10]. En outre, nous comparons également la meilleure solution visitant chacun des nœuds que nous avons trouvée à la solution optimale connue pour le problème du voyageur de commerce monoobjectif (TSP), qui est disponible sur le site de la TSPLIB [11]. Le tableau 2.11 donne le ratio d'erreur entre la meilleure solution connue et la meilleure solution que nous avons trouvée sur l'ensemble de nos expérimentations pour chacun des objectifs uniques identifiés.

Par rapport aux résultats obtenus pour le problème de Ring-Star monoobjectif, le ratio d'erreur est toujours de moins de 5% pour chacune des instances testées. L'optimum a même été trouvé pour l'instance *kroA100* avec $\alpha = 9$, et une meilleure solution a été obtenue pour l'instance *kroA200* avec $\alpha = 3$ et $\alpha = 9$. En ce qui concerne les solutions optimales du TSP, nos résultats sont assez proches pour des instances de moins de 150 nœuds. Pour les instances de plus grande taille, ils ne semblent pas aussi bons. Ainsi, en comparaison aux résultats monoobjectif optimaux ou quasi optimaux, les méthodes de recherche que nous avons proposées pour résoudre le problème de Ring-Star semblent assez prometteuses, d'autant que le temps de calcul alloué est relativement faible par rapport à la taille des instances de problème à résoudre. Rappelons toutefois que la comparaison est ici faite à l'aide des meilleurs résultats que nous avons obtenus au cours de l'ensemble des expérimentations que nous avons réalisées. Il est donc très probable que les résultats ne soient pas aussi bons pour une simulation unique.

9. Il ne nous a pas été possible de comparer nos résultats à ceux de l'autre formulation du problème de Ring-Star monoobjectif (Labbé *et al.*, 2005; Moreno Pérez *et al.*, 2003; Renaud *et al.*, 2004), pour lequel le coût d'affectation est soumis à une contrainte ; ceci en raison de la façon dont la borne a été fixée.

10. En réalité, les auteurs ont imposé un temps limite de calcul durant leurs expérimentations. Ils ont reporté la meilleure solution trouvée dans le cas où le temps de calcul dépassait ce délai, ce qui est le cas pour l'instance *kroA200* avec $\alpha = 3$, 5 et 9 dans le tableau 2.11.

11. `http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/`.

2.3.2.5 Discussion

Au cours de cette section, quatre métaheuristiques ont été appliquées à la résolution du problème de Ring-Star. Un algorithme de recherche locale, IBMOLS, a d'abord été conçu dans une variante itérative avec un voisinage multiple. Ensuite, deux algorithmes évolutionnaires multiobjectif de la littérature, à savoir l'algorithme de référence NSGA-II et un algorithme plus moderne comme IBEA, ainsi qu'un troisième, SEEA, proposé ici pour la première fois, ont tous été appliqués au cas particulier du problème traité.
Nous pouvons conclure que la version itérative de l'algorithme IBMOLS est globalement nettement plus performante que tous les autres algorithmes évolutionnaires que nous avons étudiés. Néanmoins, le comportement d'une méthode de recherche aussi simple que SEEA par rapport à des algorithmes de référence comme NSGA-II et IBEA est assez encourageant en ce qui concerne la résolution de problèmes combinatoires.
Une des principales caractéristiques du problème de Ring-Star semble être le nombre relativement élevé de points situés sur le front Pareto. Ainsi, après un certain nombre d'itérations, il est fort probable qu'une grande partie de la population courante des algorithmes I-IBMOLS, IBEA et NSGA-II ne contienne que des solutions non-dominées. Ceci pourrait expliquer la faible efficacité de NSGA-II. En effet, puisque la même valeur de fitness est affectée à la majeure partie de la population, seule la distance de *crowding* est utilisée pour confronter les solutions. Or, la technique d'affectation des valeurs de fitness basée sur un indicateur qui est utilisé par I-IBMOLS et IBEA est assurément beaucoup plus adaptée à la comparaison de solutions potentiellement Pareto optimales que la simple distance de *crowding*. Notez que ce n'est pas le cas pour SEEA car toutes les solutions non-dominées trouvées par l'algorithme peuvent prendre part au moteur d'évolution. Cependant, les bonnes performances de l'algorithme I-IBMOLS pourraient également dépendre de la proximité des solutions non-dominées dans l'espace décisionnel. Ceci est encore une fois liée à la notion de connexité (Ehrgott et Klamroth, 1997). En effet, si ces solutions sont proches les unes des autres selon l'opérateur de voisinage, les méthodes de recherche locale sont connues pour être particulièrement bien adaptées à la recherche de nouvelles solutions intéressantes.
Lors d'une prochaine étape, il serait intéressant de concevoir une stratégie de coopération entre les deux types de méthode de recherche (algorithmes évolutionnaires et algorithmes de recherche locale) afin de bénéficier simultanément de leurs caractéristiques respectives.

Conclusion

Ce chapitre s'est attardé sur l'une des contributions principales de cette thèse, à savoir la conception, l'implémentation, et l'analyse expérimentale de métaheuristiques pour l'optimisation multiobjectif. Les principaux apports de cette partie peuvent être résumés comme suit.

Conception unifiée. Un travail d'abstraction et d'unification à propos de la conception de métaheuristiques pour l'optimisation multiobjectif a été réalisé. Nous avons mis l'accent sur les questions principales liées à la convergence (et à l'affectation des valeurs de fitness), à la préservation de la diversité et à l'élitisme. De plus, une classification des stratégies existantes pour chacune de ces étapes a été proposée. Nous nous sommes ensuite concentrés sur deux types de méthodologie particuliers que nous avons étudiés de manière approfondie : les algorithmes évolutionnaires, et les algorithmes de recherche locale. Pour chacun d'eux, nous avons identifié

les principaux composants communs. Puis, nous avons proposé un modèle de conception unifié, basé sur une décomposition fine, et pour lequel bon nombre d'algorithmes existants peuvent être vus comme de simples spécifications d'un même cadre conceptuel.

Plateforme logicielle. Le cadre de conception unifiée a été utilisé comme point de départ pour la mise en œuvre d'une plateforme logicielle généraliste : ParadisEO-MOEO, module de ParadisEO. Il s'agit d'une plateforme boite blanche et orientée-objet dédiée à l'implémentation flexible et réutilisable de métaheuristiques pour l'optimisation multiobjectif. Elle est basée sur une séparation conceptuelle claire entre les méthodes de résolution et le problème qu'elles sont destinées à résoudre, conférant donc une réutilisabilité maximale du code et de la conception. Cette plateforme a été validée expérimentalement par la résolution d'une large étendue de problèmes d'optimisation multiobjectif, à la fois académiques et réels, et de différents types (continus, combinatoires).

Nouvelles approches métaheuristiques. En nous basant sur les modèles de conception unifiés présentés, nous avons également proposé de nouvelles approches méthodologiques dédiées à la résolution d'une catégorie très générale de problèmes d'optimisation multiobjectif. Tout d'abord, un algorithme évolutionnaire élitiste simple, SEEA, a été développé. Celui-ci nous a permis d'obtenir des résultats encourageants pour le problème de Ring-Star, notamment vis-à-vis de quelques algorithmes classiques du domaine, comme NSGA-II ou IBEA. Nous pensons que SEEA peut représenter une alternative intéressante à ces méthodes pour la résolution de problèmes combinatoires, en particulier lorsque le temps de calcul disponible est relativement faible. Par ailleurs, nous avons également présenté un ensemble d'algorithmes de recherche locale basés sur une relation de dominance, qui suivent tous une terminologie commune. Certains d'entre eux correspondent à des algorithmes existants ou les généralisent, d'autres correspondent à de nouvelles propositions algorithmiques. Un sous-ensemble de composants impliqués au sein de telles méthodes, leurs rôles et leurs éventuelles interactions ont été étudiés de façon approfondie lors de la résolution du problème de Flowshop à deux et à trois objectifs. Nous avons ainsi pu mettre en évidence qu'il était avantageux de dicter la stratégie d'exploration du voisinage à l'aide d'une relation de dominance, au contraire des approches couramment rencontrées de la littérature. Cette conclusion a par ailleurs été confirmée par d'autres expérimentations que nous avons menées sur le problème de voyageur de commerce multiobjectif (Liefooghe *et al.*, 2009e).

Résolution du problème de Ring-Star. Un nouveau problème de tournées, le problème de Ring-Star, a été pour la première fois étudié dans un contexte multiobjectif. En dépit de sa nature biobjectif claire, ce problème a toujours été traité de façon monoobjectif. Dans un premier temps, nous avons proposé une modélisation pour sa résolution à l'aide de métaheuristiques (représentation, évaluation, etc.). Puis, un ensemble de quatre métaheuristiques ont été adaptés à cette modélisation afin de trouver une approximation de l'ensemble Pareto optimal : IBMOLS, NSGA-II, IBEA et SEEA. Nous avons conclu que l'algorithme IBMOLS était globalement le plus performant, mais que SEEA semblait être un concurrent solide. Lors d'une prochaine étape, il pourrait donc se montrer intéressant de concevoir un schéma de coopération entre ces deux méthodes afin de bénéficier de leurs caractéristiques respectives.

Par la suite, nous allons considérer les métaheuristiques pour l'optimisation multiobjectif que nous venons d'étudier comme des agents de recherche autonomes. Au cours du chapitre suivant, celles-ci seront intégrées au sein de métaheuristiques de plus haut niveau, où elles coopéreront les unes avec les autres dans le but d'améliorer leurs performances.

Chapitre 3

Métaheuristiques coopératives pour l'optimisation multiobjectif

Dans ce chapitre, nous nous intéressons à la coopération de métaheuristiques pour l'optimisation multiobjectif. Suite à une classification des métaheuristiques hybrides, deux nouvelles approches coopératives sont proposées.

- *Une approche en mode relais durant laquelle le processus de recherche alterne entre deux agents de haut niveau : un algorithme évolutionnaire et une méthode de recherche locale. Deux variantes sont présentées, une version pour laquelle la coopération est systématique, et une autre où l'hybridation s'effectue de façon adaptative (Liefooghe* et al.*, 2010).*
- *Une approche basée sur la division de l'espace objectif à l'aide de points de référence multiples, où plusieurs agents autonomes sont exécutés en parallèle et se concentrent sur une sous-partie de l'espace objectif (Figueira* et al.*, 2009).*

Sommaire

Principales publications en rapport avec le chapitre

FIGUEIRA, J. R., LIEFOOGHE, A., TALBI, E.-G. et WIERZBICKI, A. P. (2009). A parallel multiple reference point approach for multi-objective optimization. *European Journal of Operational Research*. (à paraître).

LIEFOOGHE, A., JOURDAN, L., JOZEFOWIEZ, N. et TALBI, E.-G. (2008). On the integration of a TSP heuristic into an EA for the bi-objective ring star problem. *In International Workshop on Hybrid Metaheuristics (HM 2008)*, volume 5296 de *Lecture Notes in Computer Science*, pages 117–130, Malaga, Spain. Springer-Verlag.

LIEFOOGHE, A., JOURDAN, L. et TALBI, E.-G. (2010). Metaheuristics and cooperative approaches for the bi-objective ring star problem. *Computers & Operations Research*, 37(6):1033–1044.

Introduction

Comme remarqué lors du premier chapitre, la conception de métaheuristiques pour la résolution de problèmes d'optimisation combinatoire est généralement une question d'équilibre entre intensification et diversification. Cela est encore plus vrai dans le cadre de l'optimisation multiobjectif, où le but de la recherche est de trouver une approximation de l'ensemble Pareto optimal de bonne qualité en termes de convergence et de diversité. En règle générale, les méthodes de recherche locale sont réputées pour être particulièrement puissantes pour l'exploitation des meilleures solutions trouvées, alors que les algorithmes évolutionnaires sont plutôt efficaces pour explorer l'espace de recherche à l'aide de leurs opérateurs de variation. Plutôt que de tenter d'améliorer une méthode en termes de diversification, ou une autre en termes d'intensification, une approche attrayante consiste à hybrider les deux, ceci afin de bénéficier de leurs caractéristiques respectives. Ainsi, les métaheuristiques coopératives ont montré leur efficacité pour résoudre de nombreux problèmes d'optimisation (Talbi, 2002), dont des problèmes multiobjectif (Ehrgott et Gandibleux, 2008; Talbi, 2009). L'idée de faire coopérer différentes métaheuristiques n'est pas neuve. Rapidement, il est apparu que toutes les méthodes n'avaient pas les mêmes propriétés. On a donc cherché à profiter des avantages de ces différentes approches afin de rendre les méthodes de résolution plus performantes et plus robustes. Une métaheuristique coopérative (ou métaheuristique hybride) est une approche résultant de la combinaison de plusieurs méthodes d'optimisation (métaheuristique, heuristique, ou méthode exacte) dont au moins une métaheuristique. Ici, nous nous intéressons aux méthodes hybrides exclusivement constituées de métaheuristiques. Plus particulièrement, nous allons considérer les métaheuristiques de base, telles que celles qui ont été présentées dans le chapitre précédent, comme des méthodes de recherche autonomes s'échangeant de l'information les unes avec les autres au cours de la recherche. Par rapport au chapitre précédent, nous nous plaçons donc à un niveau d'abstraction encore plus élevée, où les métaheuristiques sont vues comme des entités propres.

Premièrement, nous allons présenter la classification de métaheuristiques hybrides proposée par Talbi (2002). Nous allons également nous attarder sur la coopération de métaheuristiques pour la résolution de problèmes d'optimisation multiobjectif, avec des exemples illustratifs de chacune des classes d'hybridation dédiées à l'optimisation multiobjectif. Ensuite, dans la section 3.2, nous proposons un schéma d'hybridation entre un algorithme évolutionnaire et un algorithme de recherche locale. L'originalité de cette première approche réside dans le fait que les étapes de recherche évolutionnaire et de recherche locale sont exécutées en phases séquentielles successives et répétées, et qu'aucune méthode de scalarisation des fonctions objectif n'est ici considérée au cours du processus de recherche. De plus, deux variantes de ce même schéma de coopération sont proposées. L'une se base sur une coopération systématique à chaque étape du processus de recherche. Alors qu'au sein de l'autre variante, l'hybridation est opérée de façon adaptative, en fonction de la vitesse de convergence de l'algorithme. Quant à elle, la section 3.3 présente une approche de coopération basée sur la division de l'espace objectif par le biais de points de référence multiples. Suite à une répartition uniforme d'un ensemble de points de référence au sein d'une région couvrant la totalité du front Pareto, plusieurs spécifications d'un même algorithme évolutionnaire sont exécutés en parallèle, dans un environnement de calcul distribué. Chacune d'elles concentre ses efforts sur une sous-région de l'espace objectif délimitée par un point de référence.

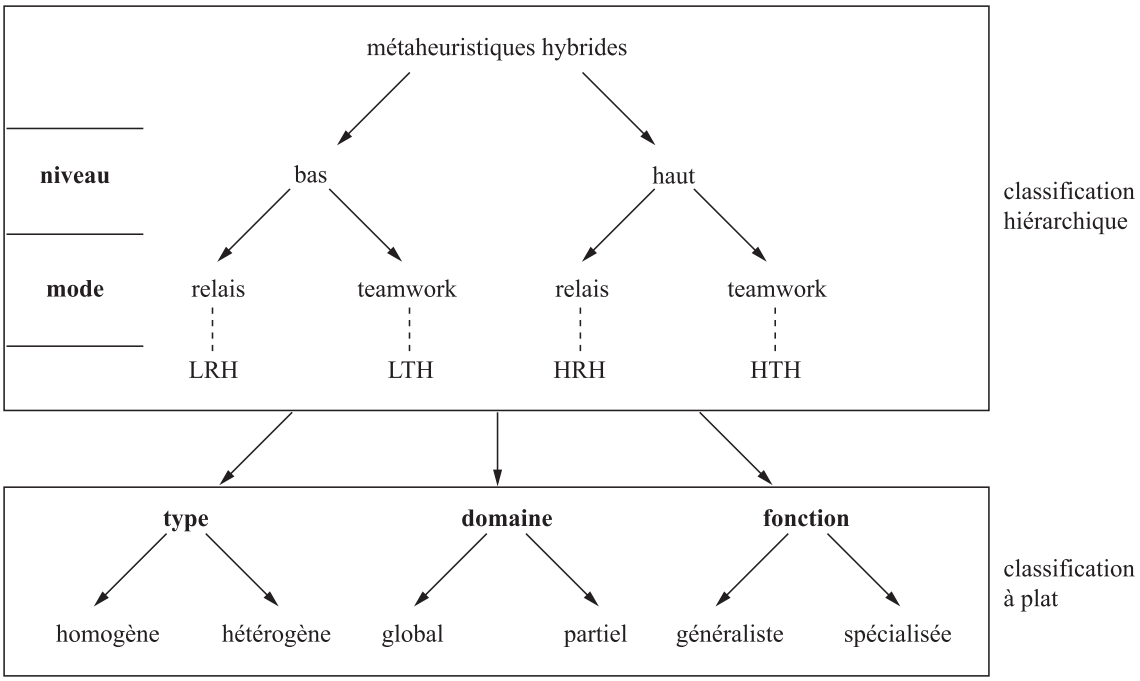


Fig. 3.1 – Taxinomie des métaheuristiques coopératives (Talbi, 2002).

3.1 Classification des métaheuristiques coopératives

De façon générale, deux vues sont nécessaires pour classer une métaheuristique coopérative (Talbi, 2002) : la *classification hiérarchique* qui traite du niveau et du mode de coopération, et la *classification à plat* qui concerne l'homogénéité des méthodes, leurs domaines, et la nature de leurs fonctions. Ces deux notions sont illustrées sur la figure 3.1 et sont détaillées ci-dessous.

3.1.1 Classification hiérarchique

Selon la taxinomie proposée par Talbi (2002), deux niveaux (bas et haut) et deux modes (relais et *teamwork*) de coopération peuvent être identifiés.
Tout d'abord, dans le cadre d'une hybridation de *bas niveau*, les éléments fonctionnels d'une métaheuristique sont modifiés. Une fonction interne d'une métaheuristique est ainsi remplacée par une autre métaheuristique. Au contraire, dans une hybridation de *haut niveau* les différentes métaheuristiques sont autonomes. Il n'existe pas de relation directe entre leurs fonctionnements internes, et les méthodes originales ne sont pas modifiées.
Dans une coopération en *mode relais*, les métaheuristiques coopératives opèrent les unes à la suite des autres dans un ordre préétabli. Chaque méthode reçoit en entrée le résultat produit par la méthode précédente, comme dans un pipeline. Le *mode teamwork* représente un modèle d'optimisation coopératif où différentes méthodes d'optimisation (des agents) évoluent en parallèle.
À partir de cette classification hiérarchique, nous obtenons quatre classes de métaheuristiques coopératives.

– **LRH (*Low-level Relais Hybrid*)**. La classe de bas niveau en mode relais regroupe les coopérations où une métaheuristique est intégrée à une métaheuristique à base de solution

unique.
- **LTH (*Low-level Teamwork Hybrid*)**. La classe de bas niveau en mode teamwork regroupe les coopérations où une métaheuristique est intégrée à une métaheuristique à base de population.
- **HRH (*High-level Relais Hybrid*)**. La classe de haut niveau en mode relais regroupe les coopérations où des métaheuristiques autonomes sont exécutées en séquence.
- **HTH (*High-level Teamwork Hybrid*)**. La classe de haut niveau en mode teamwork regroupe les coopérations où des métaheuristiques autonomes sont exécutées en parallèle et coopèrent les unes avec les autres pour résoudre le problème.

Des exemples appartenant à ces différentes classes de métaheuristiques hybrides sont données par la suite dans le cadre de l'optimisation multiobjectif.

3.1.2 Classification à plat

Selon cette même taxinomie (Talbi, 2002), la classification à plat des métaheuristiques coopératives concerne l'homogénéité des méthodes hybridées, leurs domaines d'application et leurs fonctions (Fig. 3.1).
Dans une hybridation homogène, tous les algorithmes hybridés se basent sur la même métaheuristique, comme au sein du modèle en îles. Au contraire, une hybridation hétérogène est constituée d'éléments différents.
D'un autre point de vue, on peut distinguer deux domaines de coopération : global ou partiel. Lors d'une hybridation de domaine global, toutes les métaheuristiques explorent le même espace de recherche complet. Dans un domaine partiel, le problème à résoudre est d'abord décomposé en sous-problèmes ayant chacun leur propre espace de recherche.
Enfin, un dernier critère de classification concerne la fonction de l'hybridation. Pour les coopérations de fonction généraliste, toutes les métaheuristiques s'attaquent au même problème d'optimisation. À l'inverse, les coopérations de fonction spécialisée combinent des méthodes traitant de problèmes différents.

3.1.3 Métaheuristiques coopératives et optimisation multiobjectif

En optimisation multiobjectif, la taxinomie présentée ci-dessus s'applique aisément, même si quelques ajustements mineurs sont souvent nécessaires. Ils sont dûs au fait que le but de la recherche est désormais de fournir un ensemble de solutions non-dominées, et non une solution unique.
Dans leur étude sur les métaheuristiques hybrides dédiées à la résolution de problèmes d'optimisation multiobjectif, Ehrgott et Gandibleux (2008) identifient trois catégories de méthodes d'hybridation entre un algorithme à base de population et un algorithme de recherche locale : une hybridation pour rendre une méthode plus agressive, une hybridation pour diriger une méthode, et une hybridation pour exploiter des forces complémentaires. Les deux premières catégories peuvent être vues comme appartenant à la classe LTH, la différence étant que la deuxième regroupe les hybridations dont la fonction est spécialisée. Enfin, la troisième catégorie rassemble les métaheuristiques coopératives de la classe HRH, où plusieurs méthodes (généralement deux) sont exécutées en séquence.
Les quatre classes de métaheuristiques hybrides appliquées dans le cadre de l'optimisation multiobjectif sont discutées ci-dessous.

LRH (*Low-level relais Hybrid*). Il existe très peu d'exemples de coopération LRH dans la littérature, puisque les métaheuristiques à base de solution unique ne sont pas adaptées à l'approximation de l'ensemble Pareto optimal.

LTH (*Low-level Teamwork Hybrid*). Cette classe est probablement la plus représentée, un grand nombre de métaheuristiques hybrides LTH ayant été appliquées à des problèmes d'optimisation multiobjectif. Ces méthodes consistent à coupler une métaheuristique à base de population à une métaheuristique à base de solution unique, telle qu'une recherche locale, un recuit simulé ou encore une recherche tabou, qui sont des méthodes d'optimisation efficaces en terme d'exploitation. Néanmoins, afin d'adapter la métaheuristique à base de solution unique au cas multiobjectif, ce type de méthode requiert généralement la formulation d'une préférence pour guider la recherche locale. La méthode correspondante est donc une méthode de type scalaire, où une quelconque agrégation (typiquement, une somme pondérée) a été réalisée afin de réduire le problème original en un problème monoobjectif.
Ainsi, les algorithmes mimétiques sont très populaires dans la communauté multiobjectif (Knowles et Corne, 2005). Le principe de base consiste à incorporer un algorithme de recherche locale au sein d'un algorithme évolutionnaire. Cette étape de recherche locale est généralement incluse en lieu et place de l'étape de mutation, même si elle peut également être ajoutée juste après l'étape de mutation, ou encore à chaque itération de l'algorithme général. Un exemple consiste en l'algorithme MOGLS (*multiobjective genetic local search*) proposé par Ishibuchi et Murata (1998); Jaszkiewicz (2002). Il est constitué d'un algorithme évolutionnaire multiobjectif où l'étape de mutation est remplacée par un raffinement local (Fig. 3.2). Une somme pondérée des fonctions objectif est aléatoirement définie afin d'obtenir une fonction fitness scalaire. Des solutions de la population courante sont ainsi sélectionnées selon cette valeur scalaire, puis recombinées, et enfin localement améliorées à l'aide d'une recherche de voisinage.
D'autres exemples plus évolués guident l'étape de recherche locale vers les régions de l'approximation de la frontière Pareto contenant une grande densité de points, ou vers des régions non explorées (Ehrgott et Gandibleux, 2008) ; la direction de la recherche locale étant définie à l'aide d'une fonction de scalarisation.

HRH (*High-level relais Hybrid*). Cette classe de coopération regroupe les hybridations où des métaheuristiques autonomes sont exécutées en séquence. La plupart des méthodes existantes consistent à exécuter une étape d'intensification (recherche locale, path relinking) sur une approximation obtenue à l'aide d'une métaheuristique à base de population. Par exemple, une recherche locale basée sur la dominance Pareto peut être appliquée à une approximation trouvée par un algorithme évolutionnaire multiobjectif (Talbi *et al.*, 2001). La figure 3.3 montre un exemple typique d'une telle hybridation.
D'autres stratégies consistent à initialiser la population d'une métaheuristique à base de population à l'aide de l'exécution multiple d'une méthode à base de solution unique connue pour être efficace pour une variante monoobjectif du problème. Plusieurs exemples sont donnés par Ehrgott et Gandibleux (2008). Lors de cette première phase, la résolution est basée sur une somme pondérée des fonctions objectif, et est répétée en faisant varier les poids qui leur sont associés.

HTH (*High-level Teamwork Hybrid*). Enfin, dans le cadre de l'optimisation multiobjectif, les hybridations de type HTH impliquent plusieurs métaheuristiques autonomes exécutées en

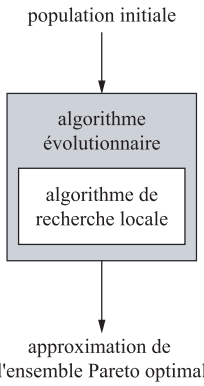


Fig. 3.2 – L'algorithme MOGLS : un exemple d'hybridation LTH.

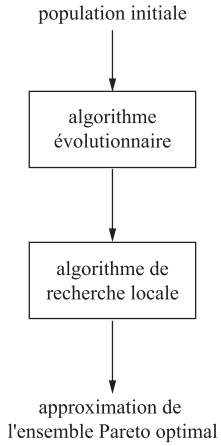


Fig. 3.3 – Un exemple typique d'hybridation HRH.

parallèle et coopérant les unes avec les autres afin de trouver une approximation de l'ensemble Pareto optimal. Cette classe d'hybridation est étroitement liée au modèle de parallélisation coopérative au niveau algorithmique, où un ensemble de métaheuristiques autonomes et parallèles échangent de l'information concernant la recherche (Talbi, 2009). Ici, le but de la parallélisation n'est pas tant d'accélérer la recherche que de trouver de meilleures solutions et de rendre la méthode globale plus robuste.

Ainsi, la finalité est de paralléliser la tâche consistant à trouver la totalité de l'ensemble Pareto optimal parmi tous les algorithmes participants. Les différentes métaheuristiques impliquées peuvent posséder différentes structures, ou encore leurs propres opérateurs ou paramètres. Par ailleurs, elles peuvent éventuellement évaluer en sous-ensemble de fonctions objectif, utiliser différents vecteurs d'agrégation pour chaque algorithme, ou diviser l'espace de recherche ou l'espace objectif en plusieurs régions. De façon générale, une ressource de calcul, le processus maître, se charge de la distribution ou de la division de la population, de l'espace de recherche, ou de l'espace objectif, ainsi que de la récupération du résultat final de chacun des algorithmes afin de construire une approximation globale de l'ensemble Pareto optimal.

Les méthodes existantes de la classe HTH se basent presqu'exclusivement sur le modèle en îles, où une métaheuristique indépendante, utilisant une population distincte, est exécutée sur chaque processeur (Fig. 3.4). Typiquement, il s'agit de plusieurs instances d'une même métaheuristique à base de population (généralement un algorithme évolutionnaire) qui évoluent en parallèle (Basseur *et al.*, 2003; Talbi et Meunier, 2006). Ces différents algorithmes coopèrent les uns avec les autres en s'échangeant régulièrement des solutions de bonne qualité au sein de leurs sous-populations (migrations). La plupart des approches basées sur le modèle en îles ne divisent pas directement l'espace (de recherche ou objectif) en différentes régions, mais résultent en la division implicite d'une population de très grande taille en sous-populations. Différentes variantes peuvent être élaborées en fonction des stratégies de communication définies entre les îles. Tout d'abord, différentes topologies d'échange d'information peuvent être définies : en anneau, en maille, en hypercube, ou à l'aide d'un graphe complet. Ensuite, ces stratégies de communication peuvent soit se baser sur les sous-populations des différentes îles, sur des archives de solutions non-dominées distribuées sur les différents nœuds, ou sur une combinaison des deux (Melab *et al.*, 2006).

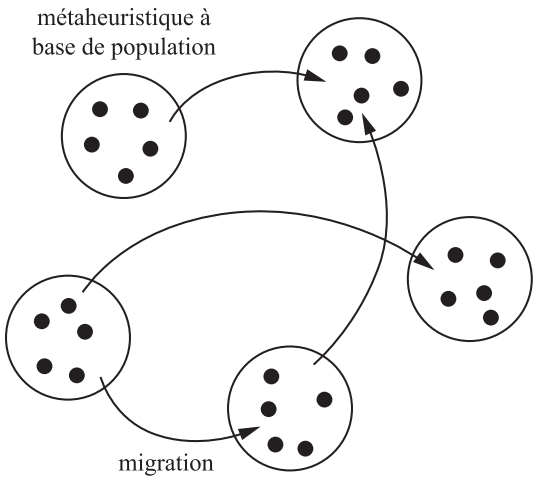


Fig. 3.4 – Le modèle en îles : un exemple d'hybridation HTH.

Aussi, une autre question essentielle à se poser lors du développement de ce type de métaheuristique hybride et parallèle concerne la gestion de l'archive (Basseur *et al.*, 2006). Celle-ci peut prendre la forme d'une *archive centralisée*, représentée à l'aide d'une structure de donnée centralisée construite au fur et à mesure du processus de recherche. Au contraire, elle peut prendre la forme d'*archives distribuées* parmi les métaheuristiques, de sorte que les algorithmes travaillent avec un ensemble local de solutions non-dominées qui devront, d'une façon ou d'une autre, être combinées à la fin de la recherche. Comme souligné par Basseur *et al.* (2006), une archive centralisée pure n'est en pratique jamais utilisée pour des questions évidentes d'efficacité. Toutes les approches centralisées de la littérature combinent en réalité des phases distribuées, où des solutions non-dominées locales sont considérées, et des phases de reconstruction de l'archive centrale.

3.2 Une approche de coopération de haut niveau en mode relais

Le premier modèle de coopération proposé repose sur une hybridation de haut niveau en mode relais (HRH). À la suite d'une courte discussion à propos des motivations d'un tel modèle, le schéma de coopération est présenté de façon générale. Enfin, une instanciation de ce schéma, où différents choix algorithmiques sont présentés, est énoncée.

3.2.1 Motivations

La finalité de l'approche proposée ici est de concevoir une collaboration entre des métaheuristiques de types différents (un algorithme évolutionnaire et un algorithme de recherche locale), ceci afin de profiter de leurs comportements respectifs. En effet, ces deux types de métaheuristique présentent des propriétés, des aptitudes et donc des qualités différentes. Un algorithme évolutionnaire offre généralement une bonne aptitude à l'exploration de l'espace de recherche. À l'inverse, un algorithme de recherche locale favorise plutôt l'intensification de la recherche. Ainsi, en gardant à l'esprit de trouver un bon compromis entre intensification et diversification lors de la conception d'une métaheuristique (surtout dans le cadre de l'optimisation multiobjectif,

où un ensemble de solutions de bonnes qualités en termes de convergence et de diversité doit être trouvé), faire coopérer ces deux types de méthode semble attractif et devrait donc nous permettre d'obtenir de meilleurs résultats que les métaheuristiques autonomes seules.
Considérons donc un algorithme évolutionnaire et un algorithme de recherche locale multiobjectif arbitraires. En règle générale, dans le domaine de l'optimisation multiobjectif, les hybridations HTH classiques se contentent d'exécuter la phase d'exploration (algorithme évolutionnaire), puis d'exploitation (algorithme de recherche locale) en séquence, avant de stopper purement et simplement le processus de recherche (Fig. 3.3). Ainsi, l'algorithme évolutionnaire fournit un ensemble de solutions non-dominées à une méthode de recherche locale qui cherche, dans un second temps, à améliorer ces solutions à l'aide d'une structure de voisinage. Or, deux observations peuvent être formulées. Premièrement, les capacités exploratrices de la première métaheuristique ne semblent pas pleinement exploitées. Il s'avérerait probablement plus intéressant d'y revenir après la phase d'intensification afin de tirer le meilleur parti d'une telle coopération ; la recherche alternant donc entre exploitation et exploration. Deuxièmement, il s'avère extrêmement difficile de déterminer préalablement le moment adéquat pour passer de la phase 1 (diversification) à la phase 2 (intensification) de la métaheuristique coopérative. Il est en général impossible de savoir à l'avance s'il vaut mieux allouer un temps de calcul équivalent entre les deux méthodes hybridées, ou s'il faut en favoriser une par rapport à l'autre.
Afin de répondre à ces deux observations, nous proposons tout d'abord d'exécuter les algorithmes évolutionnaire et de recherche locale en phases séquentielles successives et répétées. Notez que deux variantes de cette même idée sont ensuite proposées : l'une réalise une coopération systématique à chaque étape de l'algorithme hybride ; alors que dans la deuxième variante, la coopération n'est réalisée qu'en fonction de l'état actuel de la recherche, de façon adaptative.

3.2.2 Conception

3.2.2.1 Schéma de coopération

L'idée générale de notre schéma de coopération consiste à utiliser l'algorithme évolutionnaire comme processus principal, et de lancer régulièrement une méthode de recherche locale afin d'alterner la recherche entre phases d'exploration et phases d'exploitation. Étant donné qu'un grand nombre de méthodes de recherche locale possède généralement une condition d'arrêt naturelle [1], le processus de recherche évolutionnaire peut donc recommencer jusqu'à la prochaine étape de la métaheuristique hybride. Une étape peut être définie par une certaine quantité de temps de calcul, ou par un certain nombre d'itérations.
Il est important que l'algorithme évolutionnaire utilise les solutions trouvées par l'algorithme de recherche locale pour en créer de nouvelles, et vice versa. Dans le cas contraire, l'hybridation s'avérerait improductive, car les deux méthodes n'échangeraient aucune information au cours de la recherche. Ainsi, il s'avère d'abord nécessaire de spécifier une stratégie de communication d'un ensemble de solutions de l'algorithme évolutionnaire vers l'algorithme de recherche locale. Pour cela, un sous-ensemble de solutions provenant de la population principale de l'algorithme évolutionnaire, de l'archive, ou de la combinaison des deux peut être sélectionné. Il est bien évidemment préférable d'écarter une solution dont le voisinage a déjà été entièrement exploré.

1. Par exemple, comme nous l'avons vu à la section 2.1.5.1, certaines stratégies du modèle DMLS stoppent lorsque le voisinage de la totalité des solutions de l'archive a été visité. De façon similaire, l'algorithme IBMOLS s'arrête lorsque son archive interne ne reçoit plus aucune nouvelle solution (section 2.1.5.4).

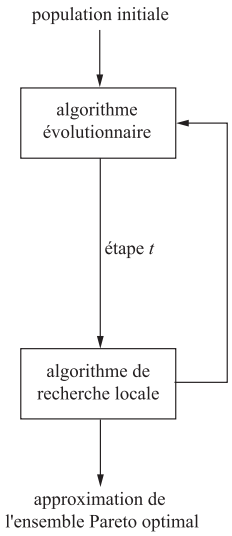


Fig. 3.5 – Une approche de coopération périodique (PCS).

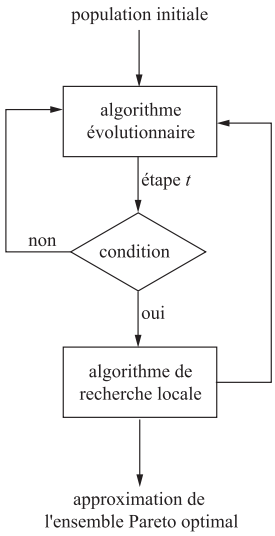


Fig. 3.6 – Une approche de coopération adaptative (ACS).

Cet ensemble de solutions constitue alors la population initiale de la phase de recherche locale. Ensuite, une fois la phase de recherche locale terminée, un nouvel ensemble de solutions a été trouvé. Plusieurs scénarios peuvent être définis pour l'intégration de ces solutions au sein du processus de recherche de l'algorithme évolutionnaire. Si l'algorithme évolutionnaire manipule une archive et que sa stratégie de sélection est élitiste, il suffit de mettre cette même archive à jour à l'aide des solutions trouvées par la recherche locale. Sinon, il s'avère toujours possible d'intégrer cet ensemble de solutions à la population principale de l'algorithme évolutionnaire, et de réduire la taille de cette dernière à l'aide d'une stratégie de remplacement quelconque.
Partant de ce principe, nous pouvons facilement imaginer deux versions de la métaheuristique coopérative : une version périodique, dans laquelle la recherche locale serait lancée à chaque étape de l'algorithme, et une version adaptative, dans laquelle la recherche locale ne serait lancée que si une certaine condition, en relation avec la vitesse de convergence de l'algorithme évolutionnaire, est vérifiée. Ces deux approches sont respectivement dénotées PCS (*periodic cooperative search*) et ACS (*adaptive cooperative search*), et sont illustrées dans les figures 3.5 et 3.6.

Ainsi, la stratégie ACS doit décider par elle-même, et en cours de recherche, s'il s'avère intéressant de lancer la recherche locale à une étape donnée du processus global. La condition que nous allons utiliser se base sur la convergence de l'approximation courante (contenue dans l'archive ou dans la population principale) d'une phase à une autre. Une possibilité est donc de mesurer la qualité de l'approximation en cours A^t en comparaison à celle de l'étape précédente A^{t-1}. Différents indicateurs existent pour évaluer la qualité d'une approximation par rapport à une autre, que ce soit en termes de convergence ou de diversité.
Nous avons vu dans la section 1.3.1 qu'il existait deux types d'indicateur de qualité : les indicateurs unaires et les indicateurs binaires. Supposons, sans perte de généralité, que les valeurs de l'indicateur choisi I soient à maximiser. Alors, dans le cas d'un indicateur binaire, il suffit de mesurer la I-valeur donnée par $\alpha = I(A^t, A^{t-1})$. Pour un indicateur unaire, on peut par exemple

calculer une valeur $\alpha = I(A^t) - I(A^{t-1})$. Ainsi, à une étape t de la stratégie de coopération, si α est inférieur à une valeur seuil δ donnée en paramètre ($\alpha \leq \delta$), alors une phase de recherche locale est lancée, sinon le processus de l'algorithme évolutionnaire continue.
Grâce à la définition du paramètre δ, il est donc possible de biaiser intentionnellement l'équilibre entre la recherche évolutionnaire et de la recherche locale au sein du modèle ACS. Cette problématique a été soulignée par Ishibuchi *et al.* (2003) dans le cadre de métaheuristiques hybrides dédiées à l'optimisation multiobjectif. De ce fait, plus la valeur de δ est élevée, plus souvent sera lancée la recherche locale. Par conséquent, la stratégie d'hybridation proposée est capable de directement traiter des spécifications différentes de l'équilibre entre recherche locale et évolutionnaire, depuis l'algorithme évolutionnaire presque pur (petite valeur de δ) vers l'algorithme de recherche locale presque pur (grande valeur de δ).

Au sein de l'approche de coopération proposée ici, le processus de recherche alterne entre deux agents autonomes de haut niveau : un algorithme évolutionnaire et un algorithme de recherche locale. Le mécanisme interne de ces deux méthodes n'est pas modifié. Par conséquent, cette métaheuristique peut être classée dans la classe LTH de la taxinomie. Aussi, cette hybridation est de type hétérogène, car les méthodes hybridées se basent sur des métaheuristiques différentes. Enfin, le domaine de l'hybridation est global et sa fonction est généraliste, car les deux algorithmes de base s'appliquent au même problème et explorent le même espace de recherche. Par rapport aux méthodes de la littérature, l'originalité de ce schéma de coopération est (*i*) de ne faire intervenir aucun mécanisme de scalarisation des fonctions objectif pour convertir le problème d'optimisation multiobjectif en un problème monoobjectif, et (*ii*) d'alterner entre recherche évolutionnaire et recherche locale, là où les méthodes existantes se contentent généralement d'exécuter ces deux méthodologies l'une à la suite de l'autre, avant de stopper le processus de recherche. En outre, l'une des variantes que nous proposons, ACS, détecte automatiquement le moment présumé idéal pour démarrer une phase de recherche locale, ceci en fonction du scénario de l'optimisation.

3.2.2.2 Choix d'instanciation du modèle

Une instanciation du schéma de coopération présenté ci-dessus requiert donc la détermination d'un algorithme évolutionnaire et d'un algorithme de recherche locale. Par la suite, en vue de l'application à venir pour la résolution du problème de Ring-Star, nous allons utiliser l'algorithme évolutionnaire SEEA et une recherche locale de type IBMOLS, ces deux derniers ayant globalement obtenu les résultats les plus prometteurs, lors d'une utilisation simple, pour le problème considéré (voir la section 2.3.2).
Concernant les stratégies de communication, les méthodes SEEA et IBMOLS maintiennent toutes deux une population secondaire (l'archive) en plus de leur population principale. Cette archive n'est pas utilisée comme simple stockage externe, mais prend également part au moteur d'évolution en servant à la construction de nouvelles solutions à explorer. Ainsi, SEEA et IBMOLS gèrent leur propre population interne et, par conséquent, utilisent l'archive comme seule mémoire partagée. La population initiale de l'algorithme IBMOLS sera donc construite à l'aide de solutions aléatoirement choisies parmi l'archive centrale. À la fin du processus de recherche locale, les solutions trouvées par IBMOLS sont tout simplement proposées comme solutions candidates pour intégrer l'archive. Avec une telle hybridation, l'archive globale est donc la seule mémoire partagée par les deux agents de recherche pour échanger de l'information.
Pour la variante ACS du modèle d'hybridation, il est également nécessaire de définir un indicateur

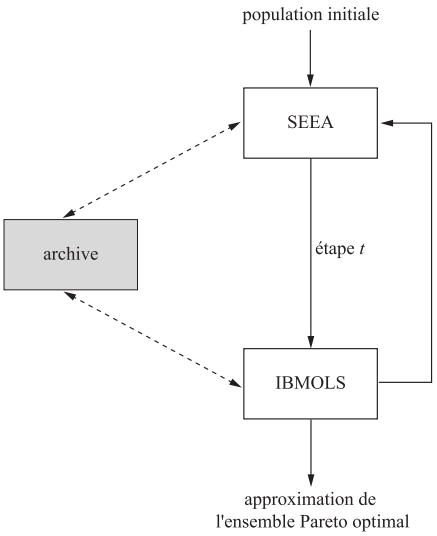


Fig. 3.7 – Instanciation du modèle PCS à l'aide des algorithmes SEEA et IBMOLS.

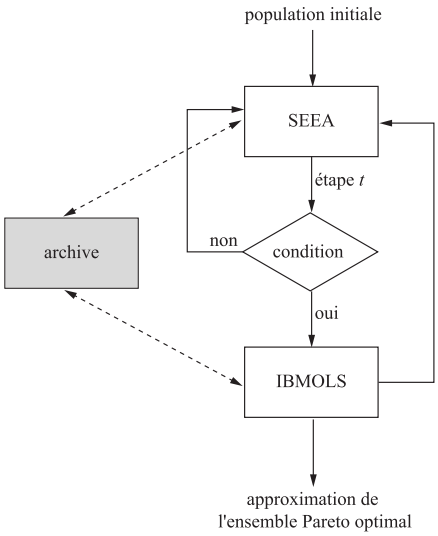


Fig. 3.8 – Instanciation du modèle ACS à l'aide des algorithmes SEEA et IBMOLS.

de qualité. Par exemple, l'utilisation de l'indicateur hypervolume aurait pu être envisagée, mais le calcul de celui-ci a l'inconvénient d'être coûteux en temps. Pour cette raison, nous allons utiliser la métrique de contribution I_C proposée par Meunier *et al.* (2000) et présentée à la section 1.3.1.4. Cette mesure donne une idée de la qualité d'une approximation par rapport à une autre en termes de convergence, et peut être calculée en un temps raisonnable. Dans notre cas, à chaque étape t de la stratégie ACS, nous allons calculer la contribution de l'archive courante A^t sur l'archive de l'étape précédente A^{t-1}. Ainsi, considérant qu'aucune solution non-dominée n'est perdue entre deux étapes, l'approximation A^t est au moins aussi bonne que A^{t-1}, de sorte que $I_C(A^t, A^{t-1}) \in [0.5, 1]$ en raison des propriétés de la métrique de contribution. En supposant que l'archive ne s'améliore plus suffisamment si la contribution de A^t sur A^{t-1} est inférieure à une valeur seuil donnée en paramètre $\delta \in [0.5, 1]$, nous choisissons de lancer le processus de recherche locale uniquement dans le cas où $I_C(A^t, A^{t-1}) \leq \delta$. Permettez-nous de remarquer qu'une stratégie ACS avec une δ-valeur inférieure à 0.5 aurait été équivalente à SEEA, et qu'une stratégie ACS avec un $\delta = 1.0$ est équivalente à PCS, le temps de calcul des différentes I-valeurs en moins.
Les métaheuristiques hybrides engendrées par l'instanciation des modèles PCS et ACS sont respectivement illustrées sur les figures 3.7 et 3.8. Elles appartiennent à la classe LTH, et peuvent donc être toutes deux dénotées par LTH(SEEA + IBMOLS) selon la grammaire proposée par Talbi (2002).

3.2.3 Implémentation

L'architecture ouverte et la flexibilité de la plateforme ParadisEO permettent de faciliter la conception et l'implémentation de métaheuristiques coopératives. Les stratégies d'hybridation les plus communes (LRH, LTH, HRH et HTH) peuvent être exploitées de façon naturelle, en particulier pour le cas de l'optimisation multiobjectif. Ainsi, les stratégies d'hybridation de bas niveau (LRH et LTH) peuvent très facilement être implémentées. Par exemple, il suffit de laisser une métaheuristique à base de solution unique prendre la place de l'opérateur de mutation d'un

TABLE 3.1 – Condition d'arrêt pour les instances additionnelles. Pour les autres instances, voir le tableau 2.7.

Instance	Temps d'exécution
pr439	50'
pr1002	50'

algorithme évolutionnaire (Boisson *et al.*, 2009). Tous les éléments sont disponibles pour réaliser une telle hybridation, même si un effort raisonnable d'implémentation reste à fournir dans le but de faciliter la transformation d'un problème multiobjectif en un problème monoobjectif, en proposant notamment de nouvelles techniques de scalarisation. Ensuite, les stratégies d'hybridation HRH, basées sur la coopération de composants de recherche indépendants, peuvent être implémentées de façon triviale sous ParadisEO. Il suffit d'exécuter séquentiellement les différents algorithmes hybridés. Par ailleurs, au sein du module ParadisEO-MOEO (Liefooghe *et al.*, 2009c,a), plusieurs classes utilitaires ont également été ajoutées, de sorte qu'il est dorénavant possible d'alterner entre deux types de méthode (par exemple, un algorithme évolutionnaire et un algorithme de recherche locale multiobjectif) ; ce qui correspond au modèle de coopération présenté dans cette section.

3.2.4 Analyse expérimentale

Ici, nous allons évaluer expérimentalement les performances des différentes variantes du premier schéma de coopération proposé (PCS et ACS) pour le problème de Ring-Star. Le design expérimental de cette partie expérimentale est suivie par quelques résultats numériques, et par une discussion à propos de la contribution de ce modèle d'hybridation pour la résolution du problème traité.

3.2.4.1 Design expérimental

3.2.4.1.1 Approches étudiées. Afin de mesurer l'efficacité de nos deux approches coopératives de haut niveau en mode relais (PCS et ACS), nous nous baserons sur la résolution du problème de Ring-Star. Nous allons les comparer aux deux méthodes basiques prenant part à leur conception ; c'est-à-dire l'algorithme évolutionnaire SEEA et l'algorithme de recherche locale IBMOLS. Bien sûr, pour des raisons évidentes d'efficacité, nous allons considérer IBMOLS dans sa version itérative : I-IBMOLS. Notez que SEEA et I-IBMOLS ont tous deux globalement obtenus les meilleurs résultats pour la résolution de la même application lors du chapitre précédent.

3.2.4.1.2 Condition d'arrêt. Tout comme les expérimentations précédemment effectuées sur le problème de Ring-Star, la condition d'arrêt se base sur un temps de calcul maximum, fixé en fonction de la taille de l'instance considérée. Néanmoins, deux instances additionnelles, de plus grande taille, seront également étudiées au cours de cette section, à savoir *pr439* et *pr1002*. Le temps maximum d'exécution pour ces deux instances est donné dans le tableau 3.1. Pour les autres instances, les valeurs correspondantes sont identiques à celles du chapitre précédent.

3.2.4.1.3 Paramètres. Pour les deux méthodes de recherche coopératives (PCS et ACS), la taille de la population gérée par SEEA est fixée à 100, et la taille de la population gérée par

IBMOLS est fixée en fonction de l'instance considérée. Ces tailles ont été établies de la même manière que pour la version itérative de l'algorithme étudié lors du chapitre précédent, voir le tableau 2.8. Cependant, pour les problèmes de grande taille, les premières expériences n'ont pas été satisfaisantes, car les algorithmes hybrides n'étaient généralement pas en mesure de lancer IBMOLS plus d'une fois durant le processus de recherche, l'exécution de ce dernier étant alors trop coûteuse en temps. Pour cette raison, nous avons limité la taille de la population d'IBMOLS à 30 solutions. Ainsi, une population de 20 individus a été choisie pour les instances de moins de 100 nœuds, et une population de 30 individus a été choisie pour les instances de 100 nœuds et plus. Pour les deux instances supplémentaires, nous avons suivi la tendance générale identifiée sur les autres instances. Nous avons donc fixé les paramètres de l'algorithme I-IBMOLS comme suit : une population de 70 individus pour *pr439* et de 100 individus pour *pr1002*, et un taux de bruit de 10%. Les autres paramètres ont été fixés de la même manière que pour les autres instances.

Une étape t considérée au sein des algorithmes hybrides a été fixé à 0.5% du temps total d'exécution. Nous avons ensuite étudié différentes valeurs de δ pour la méthode ACS : 0.6, 0.7, 0.8 et 0.9. Comme souligné précédemment, l'utilisation de la méthode ACS avec une δ-valeur inférieure à 0.5 et supérieure ou égale à 1.0 aurait été similaire à PCS et à SEEA, respectivement. Ces expérimentations préliminaires nous ont conduits à une δ-valeur efficace de 0.8.

3.2.4.2 Résultats expérimentaux

Les tableaux 3.2 et 3.3 fournissent une comparaison des résultats obtenus par SEEA, I-IBMOLS, PCS et ACS selon la métrique I_H^- et la métrique $I_{\epsilon+}^1$, respectivement. Tout d'abord, à l'exception de l'instance *eil51*, PCS et ACS sont tous deux toujours statistiquement plus performants que SEEA à l'égard d'au moins une métrique. Par rapport à I-IBMOLS, PCS et ACS obtiennent souvent de meilleurs résultats, en particulier pour les instances à 150 nœuds et plus. Et dans tous les cas, les deux méthodes hybrides ne sont jamais statistiquement surclassées par une métaheuristique non-hybride. Aussi, le bénéfice de la coopération se fait sentir sur la plupart des instances de grande taille, mais l'ajout d'un mécanisme d'adaptation ne semble pas avoir une grande influence sur les résultats. En effet, pour presque toutes les instances, les mesures n'indiquent pas de différence significative entre PCS et ACS. Les quelques exceptions notables sont *bier127* et *pr439*, où PCS obtient de meilleurs résultats par rapport à l'un ou aux deux indicateurs, et pour *pr1002* où ACS surpasse PCS selon la métrique hypervolume.

Le tableau 3.4 compare les meilleurs résultats obtenus par les méthodes hybrides sur l'ensemble de nos expérimentations et les résultats existants pour le problème de Ring-Star monoobjectif (Labbé *et al.*, 2004). Par rapport aux valeurs obtenues par les métaheuristiques basiques lors du chapitre précédent, les approches hybrides ont permis d'obtenir une amélioration du ratio d'erreur pour la quasi-totalité des instances testées. Celui-ci tourne autour de 1% pour la majorité des cas, mis à part pour les instances de très grande taille, et en particulier l'instance *pr1002* où ce ratio est proche de 15%. Mais il est vrai que, proportionnellement au nombre de nœuds, le temps d'exécution disponible pour résoudre ces instances est plus faible que pour les autres instances.

TABLE 3.2 – Comparaison des algorithmes selon l'indicateur I_H^-. Pour chaque instance, soit l'algorithme situé sur une ligne donnée domine significativement l'algorithme situé sur une colonne donnée ($\succ$), soit il est significativement dominé ($\prec$), soit il n'existe pas de différence significative entre les deux algorithmes ($\equiv$). La valeur entre parenthèses correspond à la I_H^--valeur moyenne ($\times 10^{-3}$).

			I-IBMOLS	SEEA	PCS	ACS
eil51	PCS	(4.751)	$\equiv$	$\equiv$	-	$\equiv$
	ACS	(4.482)	$\equiv$	$\equiv$	$\equiv$	-
st70	PCS	(2.691)	$\succ$	$\succ$	-	$\equiv$
	ACS	(2.865)	$\equiv$	$\succ$	$\equiv$	-
kroA100	PCS	(3.738)	$\equiv$	$\succ$	-	$\equiv$
	ACS	(3.326)	$\succ$	$\succ$	$\equiv$	-
bier127	PCS	(3.071)	$\equiv$	$\succ$	-	$\succ$
	ACS	(3.693)	$\equiv$	$\succ$	$\prec$	-
kroA150	PCS	(2.792)	$\succ$	$\succ$	-	$\equiv$
	ACS	(2.624)	$\succ$	$\succ$	$\equiv$	-
kroA200	PCS	(2.247)	$\succ$	$\succ$	-	$\equiv$
	ACS	(2.260)	$\succ$	$\succ$	$\equiv$	-
pr264	PCS	(1.342)	$\succ$	$\succ$	-	$\equiv$
	ACS	(1.404)	$\succ$	$\succ$	$\equiv$	-
pr299	PCS	(1.277)	$\equiv$	$\succ$	-	$\equiv$
	ACS	(1.293)	$\equiv$	$\succ$	$\equiv$	-
pr439	PCS	(0.348)	$\succ$	$\succ$	-	$\succ$
	ACS	(0.733)	$\succ$	$\equiv$	$\prec$	-
pr1002	PCS	(2.449)	$\succ$	$\succ$	-	$\prec$
	ACS	(0.707)	$\succ$	$\succ$	$\succ$	-

3.2.5 Discussion

Au cours de cette section, une méthode de coopération générale pour l'optimisation multiobjectif a été appliquée et expérimentée sur le problème de Ring-Star. Lors de nos expérimentations, nous avons choisi de faire coopérer l'algorithme évolutionnaire SEEA et l'algorithme de recherche locale IBMOLS. Deux variantes de cette métaheuristique coopérative ont été conçues : une variante périodique (PCS) et une variante adaptative (ACS).

En comparaison aux métaheuristiques classiques les plus performantes pour le problème considéré, les deux algorithmes hybrides ont permis une amélioration statistiquement prouvée des résultats sur un grand nombre d'instances, en particulier les instances de grande taille. Par ailleurs, en comparaison aux approches de résolution du problème de Ring-Star monoobjectif de la littérature, les algorithmes hybrides proposés pour la contrepartie biobjectif du même problème semblent fournir des solutions de bonne qualité. Toutefois, la différence d'efficacité entre PCS et ACS est presque négligeable. Ceci peut être expliqué par le fait que la stratégie ACS consacre un temps de calcul important dans le but de déterminer si la recherche locale doit être lancée ou non à chaque étape de l'algorithme. En effet, au cours de nos expériences, nous avons observé que la différence entre le nombre moyen de fois où IBMOLS est lancé au cours du processus de recherche de PCS et ACS est relativement mince. Ceci peut être expliqué par le fait que (*i*) IBMOLS consomme plus de temps de calcul à trouver des solutions intéressantes en partant d'une population de moins bonne qualité, ce qui est le cas pour PCS en comparaison à ACS (au moins pour le premier lancement), et (*ii*) une partie du temps d'exécution alloué à

TABLE 3.3 – Comparaison des algorithmes selon l'indicateur $I^1_{\epsilon+}$. Pour chaque instance, soit l'algorithme situé sur une ligne donnée domine significativement l'algorithme situé sur une colonne donnée ($\succ$), soit il est significativement dominé ($\prec$), soit il n'existe pas de différence significative entre les deux algorithmes ($\equiv$). La valeur entre parenthèses correspond à la $I^1_{\epsilon+}$-valeur moyenne ($\times 10^{-3}$).

			I-IBMOLS	SEEA	PCS	ACS
eil51	PCS	(9.561)	$\equiv$	$\equiv$	-	$\equiv$
	ACS	(9.363)	$\equiv$	$\equiv$	$\equiv$	-
st70	PCS	(6.328)	$\succ$	$\equiv$	-	$\equiv$
	ACS	(7.064)	$\equiv$	$\equiv$	$\equiv$	-
kroA100	PCS	(8.963)	$\equiv$	$\equiv$	-	$\equiv$
	ACS	(7.533)	$\succ$	$\succ$	$\equiv$	-
bier127	PCS	(8.114)	$\equiv$	$\succ$	-	$\succ$
	ACS	(9.818)	$\equiv$	$\succ$	$\prec$	-
kroA150	PCS	(5.450)	$\succ$	$\succ$	-	$\equiv$
	ACS	(5.587)	$\succ$	$\succ$	$\equiv$	-
kroA200	PCS	(5.057)	$\succ$	$\succ$	-	$\equiv$
	ACS	(5.702)	$\succ$	$\succ$	$\equiv$	-
pr264	PCS	(4.242)	$\succ$	$\equiv$	-	$\equiv$
	ACS	(4.317)	$\succ$	$\equiv$	$\equiv$	-
pr299	PCS	(4.501)	$\equiv$	$\succ$	-	$\equiv$
	ACS	(4.048)	$\equiv$	$\succ$	$\equiv$	-
pr439	PCS	(2.760)	$\succ$	$\succ$	-	$\succ$
	ACS	(5.553)	$\succ$	$\succ$	$\prec$	-
pr1002	PCS	(5.391)	$\succ$	$\succ$	-	$\equiv$
	ACS	(4.309)	$\succ$	$\succ$	$\equiv$	-

TABLE 3.4 – Ratio d'erreur entre la meilleure valeur trouvée au cours de nos expérimentations et le coût de la solution optimale du TSP monoobjectif ou le meilleur coût d'anneau trouvé pour le Ring-Star monobjectif par Labbé *et al.* (2004).

Instance	TSP	RSP monoobjectif			
	monoobjectif	$\alpha = 3$	$\alpha = 5$	$\alpha = 7$	$\alpha = 9$
eil51	0.67 %	0.67 %	0.75 %	0.37 %	0.69 %
st70	0.31 %	0.31 %	0.49 %	0.56 %	0.42 %
kroA100	0.08 %	0.08 %	0.05 %	0.12 %	**0.00 %**
bier127	0.64 %	0.64 %	0.34 %	0.52 %	0.01%
kroA150	1.38 %	1.01 %	1.04 %	0.17 %	**0.00 %**
kroA200	1.22 %	**-4.82 %**	1.05 %	0.69 %	**-1.55 %**
pr264	2.26 %	-	-	-	-
pr299	1.72 %	-	-	-	-
pr439	4.63 %	-	-	-	-
pr1002	14.51 %	-	-	-	-

ACS est utilisé pour calculer les valeurs de contribution afin de vérifier la condition d'application de la recherche locale, là où PCS consacre la totalité du temps qui lui est accordé à la recherche. Ces deux aspects ont probablement conduit au fait que le nombre de fois où IBMOLS est lancé au sein de PCS et ACS est, finalement, plus ou moins équilibré entre les deux méthodes de coopération.

3.3 Une approche de coopération de haut niveau en mode teamwork basée sur la division de l'espace objectif

Le but de la deuxième approche d'hybridation proposée est de trouver une approximation de l'ensemble Pareto optimal à l'aide de points de référence multiples. L'utilisation de plusieurs points de référence semble particulièrement bien adaptée au calcul parallèle, car les sous-problèmes résultants peuvent tous être résolus de façon indépendante.
Le principe général de l'algorithme est le suivant. Tout d'abord, un certain nombre de points de référence sont générés de façon à être uniformément répartis dans l'espace objectif. Ces points peuvent être générés à l'aide d'une approximation grossière des bornes du front Pareto, ou encore à partir de bornes inférieures et supérieures raisonnables fournies par le décideur. Une méthode de résolution est ensuite associée à chaque point de référence. Cet algorithme vise à trouver ou à approximer les solutions Pareto optimales correspondant au point de référence prédéfini (et à certains coefficients de pondération). Les méthodes de résolution basées sur un point de référence sont toutes lancées en parallèle, et un processus maître finit par fusionner les solutions non-dominées trouvées par les processus esclaves.

3.3.1 Motivations

Au cours des deux dernières décennies, une grande partie des méthodes de résolution pour l'optimisation multiobjectif proposées dans la littérature sont des approches *a posteriori*. La plupart d'entre elles consistent à trouver une approximation de l'ensemble Pareto optimal à l'aide d'un algorithme évolutionnaire. D'une part, ceci est fondé sur la conviction que la puissance de calcul des ordinateurs modernes est illimitée, et que nous pouvons les utiliser pour tout problème complexe et tout type de méthode. Cette croyance, cependant, est contredite par des expériences de calcul lors de la résolution de problèmes complexes : même le plus puissant des ordinateurs peut être facilement saturé, en raison de la dépendance non linéaire entre la complexité de calcul et la quantité de données à traiter. Ainsi, une utilisation raisonnable de la puissance de calcul existante, même si cette puissance est énorme en raison des possibilités offerte par le calcul parallèle, reste et restera probablement toujours un problème fondamental.
D'autre part, de nombreuses approches *a priori* et interactives se basent sur un point de référence où des niveaux d'aspiration et de réserve sont associés aux fonctions objectif. L'approche par point de référence a initialement été proposée par Wierzbicki (1980), et ensuite développée par de nombreux autres chercheurs (Wierzbicki *et al.*, 2000). Ces fonctions scalaires sont connues sous le nom de « *achievement scalarizing functions* ». Cette approche par point de référence résulte en la projection d'un point de référence donné (ou d'une paire de points de référence, généralement appelés points de réservation et d'aspiration) sur l'ensemble des solutions optimales. Ce point de référence représente les valeurs objectif souhaitées par le décideur. Il existe plusieurs formes d'interaction entre le décideur et l'approche par point de référence, depuis la simple modification

du point de référence jusqu'à l'utilisation d'informations supplémentaires telles que des éléments visuels basés sur des spécifications floues des valeurs de référence. Dans tous les cas, le résultat se concentre sur une région spécifique de l'espace objectif, évitant ainsi la perte de ressources de calcul en la recherche de solutions inintéressantes aux yeux du décideur.
Ici, nous proposons une nouvelle méthode combinant l'utilisation de points de référence tout en essayant d'approximer l'ensemble des solutions Pareto optimales. Au lieu d'utiliser un point de référence unique, l'idée de cette approche *a posteriori* est de définir automatiquement un ensemble de points de façon à ce que l'espace objectif soit divisé de façon uniforme, mais entièrement couvert. Chaque point donne lieu à une méthode de résolution basée sur une *achievement scalarizing function* et se concentre sur une sous-région de l'espace objectif. Ainsi, l'ensemble des solutions non-dominées peut être reconstruit en combinant les résultats de toutes les méthodes de résolution. Notez que les méthodes de résolution peuvent être lancées en parallèle puisque les problèmes d'optimisation d'une *achievement scalarizing function* peuvent être résolus de façon indépendante. L'approche parallèle à points de référence multiples proposée peut être utilisée pour résoudre des problèmes d'optimisation réels complexes, et sera par la suite appliquée au problème de Flowshop dans sa variante à deux objectifs.

3.3.2 Conception

Dans cette section, l'approche à points de référence multiples est décrite après avoir introduit quelques questions essentielles relatives à sa conception.

3.3.2.1 Éléments fondamentaux

Cette section présente quelques-uns des aspects fondamentaux liés à la conception de la méthode parallèle à points de référence multiples. Tout d'abord, un moyen d'estimer les limites du front Pareto est fourni. En second lieu, une description détaillée à propos de la façon dont les points de référence sont générés est présentée. Enfin, la méthode de résolution basée sur un point de référence unique que nous allons utiliser comme sous-processus de l'algorithme principal est donnée.

3.3.2.1.1 Achievement Scalarizing Functions. La notion d'*achievement scalarizing fonctions* proposée par Wierzbicki (1980) est souvent utilisée pour résoudre des problèmes d'optimisation multiobjectif. Cette technique se montre particulièrement bien adaptée pour fonctionner avec un point de référence. Un point de référence traduit les valeurs souhaitées ou acceptées par le décideur pour chacune des fonctions objectif. Ces valeurs objectif sont appelées *niveaux d'aspiration* et le vecteur objectif correspondant est appelé *point de référence*. Ce dernier peut être défini soit dans la région réalisable, soit dans la région non réalisable de l'espace objectif. L'une des familles d'*achievement scalarizing fonctions* peut être définie ainsi :

$$\sigma(z, z^0, \lambda, \rho) = \max_{i=1,2,\ldots,n} \left\{ \lambda_i (z_i - z_i^0) \right\} + \rho \sum_{j=1}^{n} \lambda_j (z_j - z_j^0) \tag{3.1}$$

où σ est une application de Z dans $\mathbb{R}$, $z = (z_1, z_2, \ldots, z_n)$ est un vecteur objectif, $z^0 = (z_1^0, z_2^0, \ldots, z_n^0)$ est un point de référence, $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_n)$ un vecteur de coefficients de pondération, et ρ est une petite valeur positive arbitraire ($0 < \rho \ll 1$). Le mot famille est ici utilisé

pour indiquer que plusieurs fonctions peuvent être construites selon la variabilité des coefficients de pondération et du point de référence. Maintenant le problème suivant peut être construit, où X représente l'ensemble des solutions réalisables dans l'espace décisionnel et $z = f(x)$.

$$\begin{array}{rl} \min & \sigma(z, z^0, \lambda, \rho) \\ \text{tel que :} & x \in X \end{array} \tag{3.2}$$

Pour un point de référence donné z^0, deux propriétés ont été prouvées par Steuer (1986) :
- si $x^\star = arg\min_{x \in X} \sigma(z, z^0, \lambda, \rho)$, alors $x^\star$ est une solution Pareto optimale ;
- si $x^\star$ est une solution Pareto optimale, alors il existe une fonction $\sigma(z, z^0, \lambda, \rho)$ telle que $x^\star$ est un optimum (global) du problème donné en (3.2).

Notez que les vecteurs de coefficients de pondération peuvent être normalisés, et que l'ensemble des vecteurs de coefficients de pondération normalisés peut être représenté comme suit :

$$\Lambda = \left\{ \lambda = (\lambda_1, \lambda_2, \ldots, \lambda_n) \ \mid \ \sum_{j=1}^{n} \lambda_j = 1, \ \ \lambda_j > 0, \ j = 1, 2, \ldots, n \right\}$$

Utiliser différents vecteurs λ pour un même point de référence z^0 peut conduire à différentes solutions optimales du problème défini en (3.2). Même si cela n'est pas l'objet de l'approche proposée ici, cet aspect peut être utilisé afin de concevoir un algorithme parallèle portant sur un même point de référence et sur différents vecteurs de pondération, et donc définir un deuxième niveau de parallélisme.

3.3.2.1.2 Estimer les bornes du front Pareto. Comme indiqué dans le premier chapitre, le calcul du point nadir z^n est plutôt une tâche difficile, et encore plus pour des problèmes multiobjectif discrets de grande taille. En outre, le calcul exact du point idéal z^* est généralement impossible dès que l'on traite de problèmes de grande taille. Toutefois, au sein de l'approche proposée, ces deux points sont nécessaires afin de produire l'ensemble des points de référence utilisés par l'algorithme. Nous allons donc essayer de trouver une approximation de bonne qualité de ces deux points. Ces approximations seront désignées par z^{n^A} et z^{*^A}, respectivement.
Permettez-nous de considérer deux options légitimes pour accomplir une telle tâche. D'une part, si le décideur est en mesure de fournir des bornes supérieures et inférieures raisonnables pour chaque fonction objectif, le calcul des points idéal et nadir, ou de leurs approximations, se résume à une étape initiale de mise à l'échelle. De notre expérience, il est toujours envisageable de demander au décideur de fournir des connaissances initiales sur le problème à résoudre. En outre, notez que les métaheuristiques ne peuvent généralement pas s'exécuter de façon entièrement automatique sans information initiale. Par exemple, au moins la ou les solutions de départ, même si elles peuvent être fournies par un générateur de nombres aléatoires, requièrent généralement une échelle initiale.
D'autre part, si le décideur n'est pas en mesure de fournir cet élément d'information, une autre option consiste à trouver une approximation de ces points par le biais d'une méthode de recherche dédiée, et exécutée durant un nombre relativement faible d'itérations. De façon générale, les algorithmes évolutionnaires multiobjectif donnent des approximations discrètes du front Pareto ; le point idéal et le point nadir d'une telle approximation peuvent facilement être calculés. La difficulté d'estimer le point nadir est liée au problème d'origine et au fait que de telles approximations évolutionnaires pourraient trouver une mauvaise approximation du front Pareto à proximité

du point nadir. Par conséquent, il est important d'appliquer au moins les deux aspects d'un algorithme évolutionnaire multiobjectif. Le premier consiste à utiliser une stratégie encourageant la diversité (voir la section 2.1.3.2), ou, de façon générale, renforçant l'importance des valeurs extrêmes de l'ensemble discret d'approximation ; il existe plusieurs stratégies de ce type (Szczepański et Wierzbicki, 2003). Une autre consiste à utiliser un critère d'arrêt qui concerne le diamètre de l'approximation du front Pareto, basée sur la distance entre les points idéal et nadir. Si ce diamètre se stabilise durant un nombre subséquent d'itérations, cela indique que, soit l'approximation est déjà de bonne qualité (car une bonne approximation du point nadir est l'un des aspects les plus difficiles de l'optimisation multiobjectif), soit l'algorithme n'est plus en mesure de produire une amélioration (ce qui peut toujours survenir avec des heuristiques). Szczepański et Wierzbicki (2003) fournissent un bon aperçu de divers exemples. Une telle approximation doit être corrigée en l'élargissent par un certain facteur. Cet élargissement est nécessaire pour de nombreuses raisons, à commencer par le fait que l'algorithme évolutionnaire multiobjectif se rapproche de la distance idéal-nadir par le bas, et un tel élargissement est également utilisé lors de l'exécution de méthodes linéaires multiobjectif classiques.
Par exemple, afin de trouver une approximation des points idéal et nadir, Deb *et al.* (2006) ont proposé un algorithme basé sur une modification de NSGA-II (Deb *et al.*, 2002), et qui se concentre sur la recherche vers les extrémités du front Pareto. Ensuite, les valeurs supérieures et inférieures de chaque fonction objectif sont calculées à partir de l'approximation finale du front Pareto, et peuvent donc être utilisées comme estimations des points idéal et nadir, z^{*^A} et z^{n^A}. Pour ce faire, cet algorithme utilise une stratégie de maintien de la diversité qui se base sur une distance de crowding modifiée mettant l'accent sur l'importance des moins bonnes solutions selon chaque fonction objectif.

3.3.2.1.3 Générer de multiples points de référence. Il peut exister plusieurs façons de générer de multiples points de référence. La plus intuitive est peut-être la génération aléatoire dans une zone prédéfinie de l'espace objectif. Cette façon simple, cependant, ne semble pas suffisante pour notre cas. La génération aléatoire pourrait ne pas couvrir la zone prédéfinie. C'est la raison pour laquelle nous proposons de définir une distribution uniforme des points sur une zone prédéfinie. Après avoir défini les limites de chaque fonction objectif, la zone est construite de sorte qu'elle couvre l'ensemble de la région réalisable de l'espace objectif. La figure 3.9 illustre la façon de procéder lorsque deux fonctions objectif sont considérées. Supposons, par souci de simplicité, que les bornes du front Pareto peuvent être définies de façon précise. Une façon de procéder consiste à suivre les étapes suivantes.

1. Calculer le point idéal z^* ou une bonne approximation z^{*^A}.
2. Calculer le point nadir z^n ou une bonne approximation z^{n^A}.
3. Considérer la zone formée par ces points, et définir un plan de coupe comme la « diagonale » de cette zone.
4. Si nécessaire, relâcher les points extrêmes d'un tel plan de coupe de sorte à obtenir le plan de coupe utilisé pour générer les points de référence ; cette étape est réalisée afin de ne pas utiliser les points extrêmes comme points de référence, puisqu'ils sont non-dominés.
5. Générer uniformément les points de référence en fonction du nombre de processeurs disponibles.

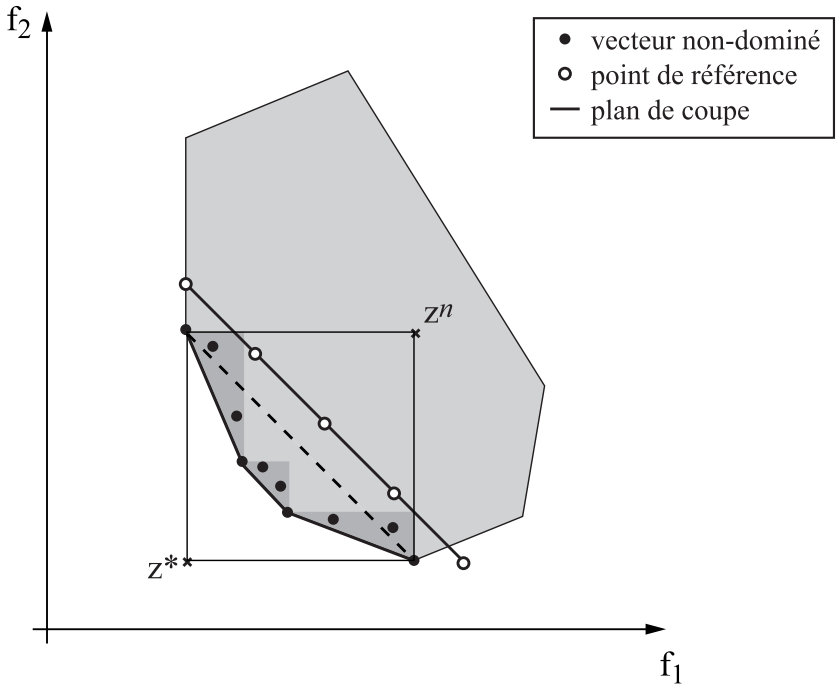


Fig. 3.9 – Plan de coupe pour un problème à deux objectifs. Dans cet exemple, cinq points de référence sont générés.

La figure 3.9 illustre la façon de définir ce plan de coupe. Le rectangle intégré a été construit à partir de z^* et z^n. La diagonale représentée par la ligne reliant ces deux sommets peut être utilisée comme premier plan de coupe. Ensuite, nous relâchons cette diagonale afin d'obtenir celle ayant un style solide. Ceci est réalisé afin d'éviter l'utilisation de points extrêmes comme points de référence, car ceux-ci sont non-dominés. Évidemment, lorsque l'on traite des approximations des points idéal et nadir, il n'est pas nécessaire de procéder à une telle modification. Dans le cas de modèles linéaires d'optimisation multiobjectif continue, le plan couvre la totalité de la région non-dominée qui est au-dessus de lui-même. Pour un problème discret, ce n'est pas toujours le cas, comme nous pouvons l'observer sur la figure. Il existe des points non-dominés au-dessus et en dessous de la ligne de coupe. Cela ne pose pas de problème puisque l'*achievement scalarizing function* définie en (3.1) n'impose pas de restriction sur l'emplacement des points de référence. Une telle conclusion n'est pas vraie lorsqu'une métrique de Chebychev est utilisée, comme c'est le cas dans de nombreux travaux de recherche du domaine de la prise de décision interactive.
La figure 3.10 illustre comment définir les points de référence initiaux pour un problème à trois objectifs. Dans un tel cas, le domaine est symbolisé par un triangle dans lequel est représentée une grille de quinze points de référence. Ce principe peut être facilement généralisé pour des problèmes à n dimensions, mais le nombre requis de points de référence croît de façon exponentielle par rapport à n.

3.3.2.1.4 Un algorithme évolutionnaire à point de référence unique. Dans ce paragraphe, nous présentons un algorithme basé sur la notion d'*achievement scalarizing functions* et conçu pour être utilisé avec un seul point de référence z^0 et un vecteur de coefficients de

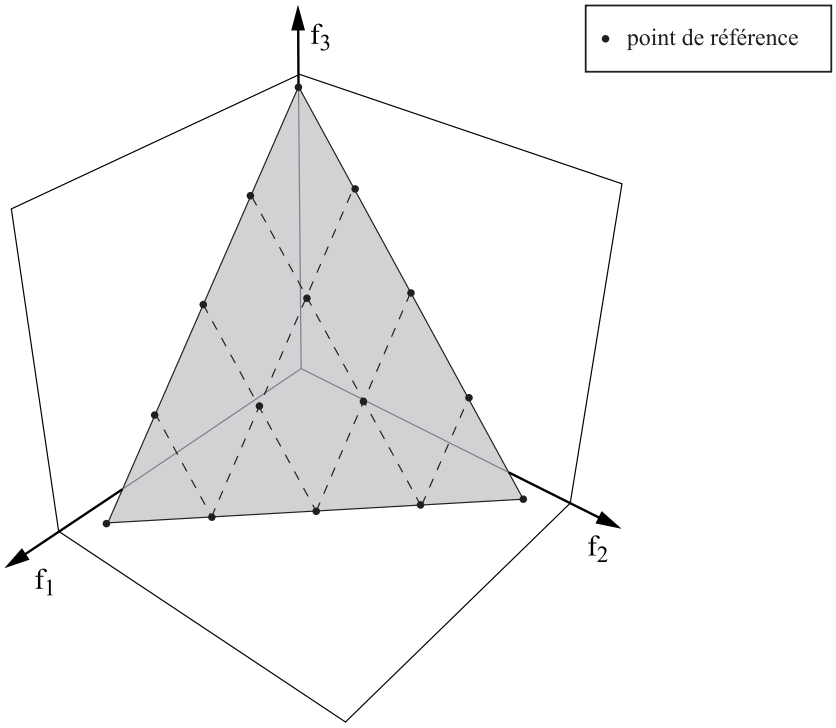


Fig. 3.10 – Illustration de la création des points de référence pour un problème à trois objectifs.

pondération λ. Cette approche à point de référence unique sera utilisée comme sous-procédure de l'algorithme principal dans le but de résoudre le problème résultant de z^0 et λ.
Traditionnellement, l'utilisation d'*achievement scalarizing functions* permet de transformer le problème d'optimisation multiobjectif original en un problème monoobjectif, pour lequel une solution unique est alors obtenue (Fig. 3.11). Une telle métaheuristique, visant à obtenir une solution unique correspondant à une formulation du problème donné en (3.2), a par exemple été proposée par Szczepański et Wierzbicki (2003). Néanmoins, dans notre cas, nous recherchons plutôt un ensemble de solutions, situées dans la zone de projection d'un point de référence unique. Par ailleurs, comme souligné par Molina *et al.* (2009), dans la plupart des cas, une méthode interactive est généralement requise pour obtenir la solution préférée du décideur, car ce dernier acquiert de la connaissance à propos de ses préférences et du problème à résoudre tout au long du processus d'interaction ; ceci résultant en une modification du point de référence ou des poids au cours de la recherche. Ainsi, une projection du front Pareto aux alentours de la solution projetée semble plus appropriée que seule la solution projetée, car un plus grand nombre d'alternatives sont alors proposées, chacune étant *a priori* adaptée aux préférences du décideur (Fig. 3.12).

Très peu de métaheuristiques à base de population, et donc adaptées à la recherche d'un ensemble de solutions, ont été proposées dans ce but. On peut par exemple citer les travaux de Deb et Sundar (2006) ou de Molina *et al.* (2009). Ici, nous avons choisi de considérer l'algorithme évolutionnaire multiobjectif à base de préférence (*preference-based evolutionary algorithm*, PBEA) proposé par Thiele *et al.* (2009). PBEA vise à produire un ensemble de bonne qualité, probablement de petite taille, de solutions non-dominées associées à un point de référence z^0 et à un

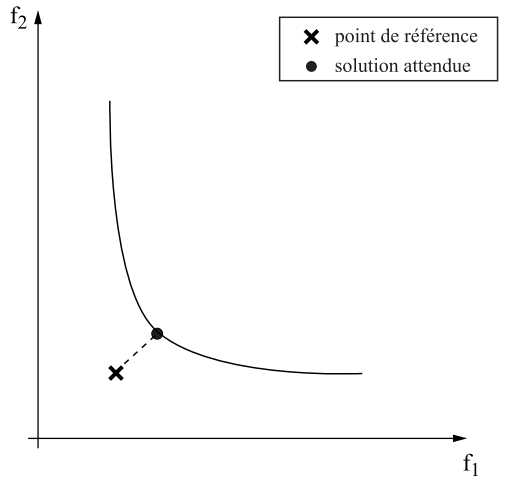


Fig. 3.11 – Projection unique sur le front Pareto.

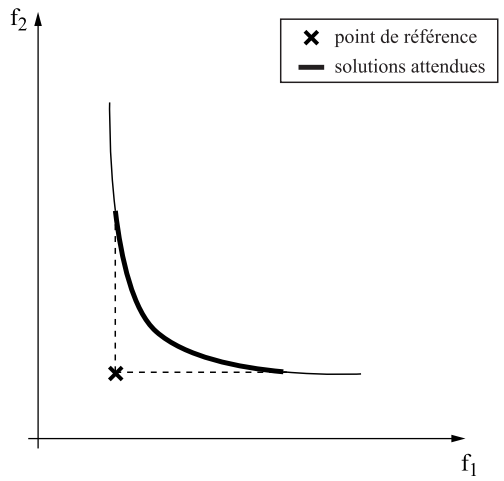


Fig. 3.12 – Projection d'un ensemble de solutions sur le front Pareto.

vecteur de coefficients de pondération λ.
PBEA est une variante de l'algorithme IBEA (Zitzler et Künzli, 2004) où l'information liée aux préférences est prise en compte par l'intermédiaire d'un point de référence. Il est basée sur l'indicateur epsilon additif ($I_{\epsilon+}$), couramment utilisé au sein d'IBEA, et sur les *achievement scalarizing functions* données dans la définition 3.1. Tout d'abord, une version normalisée des *achievement scalarizing functions* est définie.

$$\sigma(f(x), z^0, \lambda, \rho, \delta) = \sigma(f(x), z^0, \lambda, \rho) + \delta - \min_{x' \in P}\{\sigma(f(x'), z^0, \lambda, \rho)\} \tag{3.3}$$

Le paramètre $\delta > 0$ donne une *spécificité* représentant la valeur minimale de la fonction normalisée. Il permet de spécifier la largeur de l'amplification donnée par l'indicateur epsilon pour les solutions proches du point de référence. Ainsi, un indicateur de qualité basé sur une préférence (I_p) peut être défini.

$$I_p(x, x') = I_{\epsilon+}(x', x)/\sigma(f(x), z^0, \lambda, \rho, \delta) \tag{3.4}$$

Cet indicateur de qualité peut maintenant être utilisé au sein de la stratégie d'affectation des valeurs de fitness de l'algorithme IBEA présenté lors de la section 2.1.4.1, en lieu et place de l'indicateur $I_{\epsilon+}$. L'algorithme résultant est dénommé PBEA.
Afin de donner approximativement la même amplitude à chaque fonction objectif, nous avons légèrement modifié l'algorithme PBEA original de la façon suivante. Chaque élément z_i d'un vecteur objectif z est remplacé par une valeur normalisée z'_i à l'aide des points idéal et nadir (ou de leurs approximations) ; points que nous avons obtenus lors de la phase initiale de l'algorithme et éventuellement affinés au cours de la recherche. Pour $i \in \{1, \ldots, n\}$:

$$z'_i = \frac{z_i - z_i^\star}{z_i^n - z_i^\star} \tag{3.5}$$

De plus, une archive additionnelle a été ajoutée à PBEA afin de stocker l'approximation courante. Celle-ci est mise à jour à chaque itération de l'algorithme. Chaque algorithme à point de référence unique possède donc sa propre archive locale.

3.3.2.2 Schéma de coopération

L'approche parallèle à points de référence multiples proposée peut être divisée en trois phases consécutives. La première, appelée phase de préparation, est consacrée à la conception de plusieurs versions d'un algorithme à point de référence unique en estimant les limites du front Pareto, en générant de multiples points de référence, et en concevant une version de l'algorithme PBEA pour chaque point de référence. La deuxième phase est la phase de recherche et consiste à lancer les algorithmes PBEA correspondant à chaque point de référence en parallèle, jusqu'à ce qu'une condition d'arrêt soit satisfaite. Enfin, la phase de finalisation vise à fusionner les solutions non-dominées trouvées par les sous-procédures afin de construire une approximation complète de l'ensemble Pareto optimal. Ces trois phases sont détaillées ci-dessous.

3.3.2.2.1 Phase de préparation. La phase de préparation est consacrée à attribuer un nombre égal de processeurs et de points de référence. Elle se compose des étapes suivantes.

1. Obtenir une estimation des bornes inférieures et supérieures du front Pareto. Cette estimation peut être fournie par le décideur, ou approximée automatiquement. Dans ce dernier cas, un critère d'arrêt possible peut être défini comme suit. Tout d'abord, déterminer le diamètre extérieur de l'approximation courante ; c'est-à-dire la norme de Chebychev de la distance entre les bornes inférieure et supérieure. Ensuite, comparer cette valeur avec l'estimation du diamètre extérieur de l'itération précédente. Si la différence entre ces deux valeurs reste constante pendant un nombre d'itérations donné en paramètre, arrêter la recherche.
2. Déterminer un plan de coupe à travers la zone (normalisée) des valeurs objectif, et définir une répartition uniforme de m points de référence dans cette zone.
3. Pour chaque point de référence z^0, concevoir une version de PBEA liée à l'*achievement scalarization function* définie par z^0 et un vecteur de coefficient $\lambda = (1/n, 1/n, \ldots, 1/n)$, où n est le nombre de fonctions objectif.

3.3.2.2.2 Phase de recherche. La phase de recherche consiste tout simplement à exécuter les m versions de PBEA en parallèle jusqu'à ce qu'une condition d'arrêt soit vérifiée. La valeur de m peut être définie en fonction du nombre de processeurs disponibles. Nous voyons bien ici l'intérêt d'utiliser un algorithme comme PBEA, qui permet d'obtenir un ensemble de solutions, et donc d'éviter que le nombre de solutions non-dominées obtenues par l'algorithme général soient inférieur ou égal ou nombre de processeurs utilisés.

3.3.2.2.3 Phase de finalisation. Cette ultime phase consiste à unifier les approximations trouvées par chaque algorithme à point de référence unique afin d'obtenir une approximation globale de l'ensemble Pareto optimal.

La procédure générale de l'algorithme PMRPEA (*parallel multiple reference point evolutionary algorithm*) est donnée dans l'algorithme 8. Elle est également illustrée sur la figure 3.13.

3.3.3 Implémentation

Les stratégies d'hybridation HRH se basent sur l'exécution parallèle de métaheuristiques, et requièrent donc la définition de concepts avancés de calculs parallèles et distribués. Or, le module

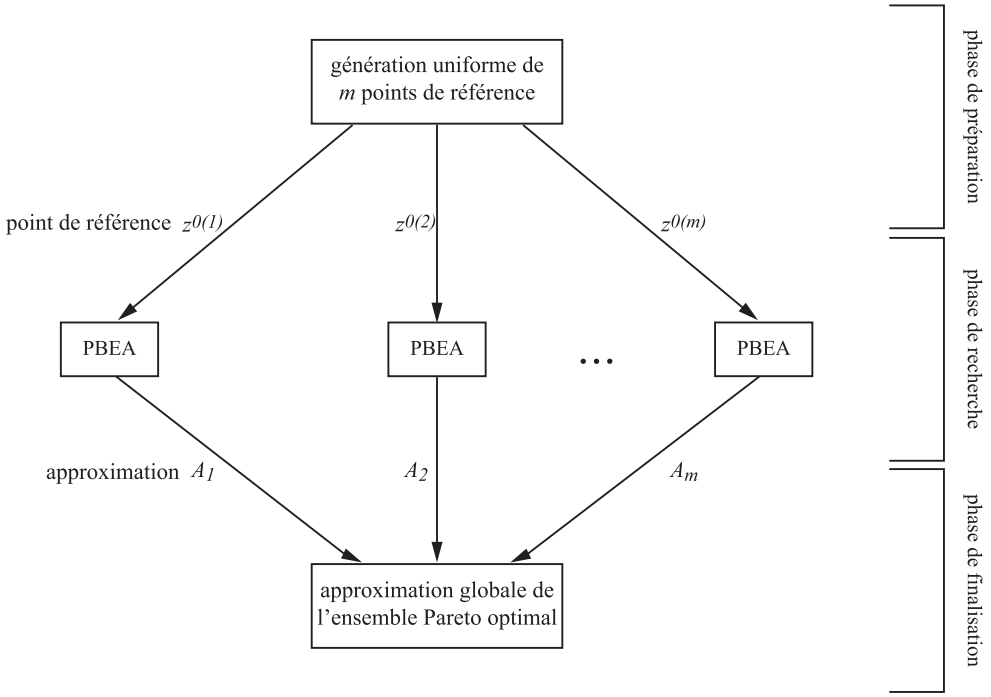


Fig. 3.13 – Illustration des étapes générales de l'algorithme PMRPEA.

Algorithme 8 Modèle de l'algorithme PMRPEA

1. Démarrer avec une approximation de la limite des valeurs objectif donnée par l'utilisateur ; utiliser cette estimation initiale des limites du front Pareto pour déterminer les approximations initiales du point idéal et nadir, z^{*^A} et z^{n^A}.
2. Soit $d^\star$ le diamètre extérieur de ces limites, mesuré par $||z^{n^A} - z^{*^A}||$.
3. Soit $\lambda = (1/n, 1/n, \ldots, 1/n)$, où n est le nombre de fonctions objectif.
4. Générer m points de référence $z^{0(1)}, z^{0(2)}, \ldots, z^{0(m)}$ de façon uniforme en utilisant les bornes (approximatives) du front Pareto.
5. Générer m populations initiales $P_1, P_2, \ldots, P_m$ de taille N.
6. Pour $i \leftarrow 1$ à m, exécuter en parallèle (sur m processeurs) :
   $A_i \leftarrow \text{PBEA}(P_i, \text{G}, z^{0(i)}, \lambda, \rho)$
   jusqu'à ce qu'une condition d'arrêt soit vérifiée.
7. Retourner les solutions non-dominées de $A_1 \cup A_2 \cup \ldots \cup A_m$.

ParadisEO-PEO offre une implémentation transparente des différents modèles de parallélisation sur différentes architectures matérielles et logicielles (Talbi, 2009). Ces modèles parallèles peuvent être conçus de façon transparente par l'utilisateur, par simple instanciation. En particulier, ParadisEO-PEO fournit une implémentation simple de modèles parallèles au niveau algorithmique, où plusieurs algorithmes coopèrent et échangent des données de n'importe quel type. À notre connaissance, ParadisEO, par le biais des module ParadisEO-MOEO et ParadisEO-PEO, est la seule plateforme logicielle permettant d'implémenter un tel schéma d'hybridation parallèle de métaheuristiques pour l'optimisation multiobjectif. Ce module a donc été utilisé lors de l'implémentation du deuxième modèle de coopération précédemment proposé. Dans notre cas, l'implémentation de cet algorithme parallèle se base sur la librairie MPI (Message Passing Implementation). MPI est un environnement de programmation parallèle permettant un déploiement portable sur des réseaux de stations de travail hétérogènes.

3.3.4 Analyse expérimentale

Dans cette section, le modèle d'hybridation basée sur la résolution parallèle à points de références multiples, est appliqué au problème de Flowshop à deux objectifs. Les différentes approches testées sont d'abord présentées et sont suivies par le protocole d'expérimentation, les résultats obtenus, ainsi qu'une discussion à propos de l'approche proposée.

3.3.4.1 Design expérimental

3.3.4.1.1 Approches étudiées. Afin de mesurer l'efficacité de l'approche proposée, nous allons comparer son comportement aux algorithmes NSGA-II (Deb *et al.*, 2002) et IBEA (Zitzler et Künzli, 2004). Le premier est plus que couramment utilisé comme référence lors d'une comparaison d'approches dédiées à l'optimisation multiobjectif. IBEA a quant à lui été choisi car PMRPEA est composé d'une méthode de résolution liée à un point de référence unique, PBEA, qui n'est autre qu'une variante de l'algorithme IBEA. Ainsi, nous espérons que l'impact de la stratégie d'hybridation sera mesuré de façon impartiale au niveau de l'implémentation.

3.3.4.1.2 Caractéristiques du réseau informatique. Les expérimentations ont été conduites sur un cluster de 2 nœuds, chacun d'eux étant composé de 2 processeurs Xeon Dual Core 5060 (3.2 GHz, 2×2 MB, 4 Gb RAM). Ainsi, un nombre total de 16 processeurs inter-connectés dans un environnement de calcul distribué étaient disponibles.

3.3.4.1.3 Paramètres. Les valeurs des paramètres utilisés au cours de nos expériences sont résumées dans le tableau 3.5. La condition d'arrêt a été motivée par le fait que, après 5000 générations sans aucune amélioration, nous pouvons raisonnablement penser que l'algorithme évolutionnaire ne convergera plus. Toutefois, une itération non améliorante est difficile à définir dans le cas multiobjectif. Ici, nous déclarons que l'itération est non améliorante si aucune nouvelle solution non-dominée n'a été incluse au sein de l'archive. En outre, en dehors de cette première condition d'arrêt, une exécution maximum de 30 minutes a été autorisée.

TABLE 3.5 – Paramètres.

Paramètre	Valeur
Nombre de points de référence	10
Taille de la population	100
Nombre maximum d'itérations sans amélioration	5000
Temps d'exécution maximum	30 minutes
Taux de croisement	0.8
Taux de mutation	1.0
κ (PBEA et IBEA)	0.05
δ (PBEA)	10^{-2}
ρ (PMRPEA)	10^{-4}

3.3.4.2 Résultats expérimentaux

Les résultats obtenus sont résumés dans les tableaux 3.6, 3.7 et 3.8. PMRPEA désigne l'algorithme évolutionnaire parallèle à points de référence multiples. Selon le protocole expérimental défini ci-dessus, la première série d'expériences a montré que les résultats obtenus par PMRPEA sont nettement meilleurs que ceux de IBEA et NSGA-II sur l'ensemble des instances considérées, et ceci selon les deux indicateurs de performance considérés (hypervolume et epsilon, tous deux à minimiser). Toutefois, comme on peut le voir dans le tableau 3.6, le temps d'exécution moyen de PMRPEA est toujours supérieur à celui des autres algorithmes étudiés. C'est la raison pour laquelle nous avons réalisé une seconde série d'expériences où le processus de recherche de PMRPEA a été stoppé après un plus petit temps de calcul. L'algorithme correspondant est désigné par PMRPEA-2. Pour une instance donnée, le temps de calcul maximum alloué à PMRPEA-2 a été fixé à la valeur moyenne minimale observée lors des expériences précédentes sur l'ensemble des autres algorithmes. Malgré ce temps de calcul moindre, PMRPEA-2 a tout de même obtenu de meilleurs résultats, selon les métriques I_H^-- et $I_{\epsilon+}$, que les algorithmes IBEA et NSGA-II ; et ceci sur toutes les instances que nous avons testées. En outre, PMRPEA-2 est toujours précédé de PMRPEA, à l'exception des instances de grande taille à 100 jobs, où la différence entre les deux n'est pas significative, voir les tableaux 3.7 et 3.8. Mais il est vrai que pour ces instances, la différence entre les temps de calcul de PMRPEA et PMRPEA-2 semble proportionnellement inférieure.

3.3.5 Discussion

Les résultats de nos tests montrent que l'hybridation et la parallélisation d'algorithmes évolutionnaires multiobjectif basés sur de multiples points de référence s'avèrent très prometteuses. En effet, un tel algorithme nous a permis d'obtenir une amélioration statistiquement significative des résultats par rapport à des algorithmes évolutionnaires multiobjectif, non hybrides et non parallèles, comparables. En effet, non seulement l'approche proposée (PMRPEA) a obtenu de meilleurs résultats à temps de calcul équivalent, mais celle-ci continue de converger vers une meilleure approximation de l'ensemble Pareto optimal, là où les algorithmes classiques n'en semblent plus capables.

TABLE 3.6 – Comparaison des algorithmes selon le temps de calcul moyen, la I_H^--valeur moyenne et la $I_{\epsilon+}$-valeur moyenne.

Instance	Algorithme	Temps d'exécution moyen (secondes)	I_H^--valeur moyenne ($\times 10^{-1}$)	$I_{\epsilon+}$-valeur moyenne ($\times 10^{-1}$)
$020 \times 05 \times 01$	PMRPEA	311.1	0.942	1.582
	PMRPEA-2	111.0	0.967	1.618
	IBEA	111.5	1.905	2.046
	NSGA-II	124.9	1.722	2.000
$020 \times 10 \times 01$	PMRPEA	345.0	0.212	0.492
	PMRPEA-2	147.0	0.229	0.508
	IBEA	147.4	0.912	1.090
	NSGA-II	182.6	0.760	1.035
$020 \times 20 \times 01$	PMRPEA	510.0	0.493	0.852
	PMRPEA-2	210.0	0.514	0.881
	IBEA	210.7	1.900	2.446
	NSGA-II	218.1	2.023	2.532
$050 \times 05 \times 01$	PMRPEA	517.1	0.740	0.738
	PMRPEA-2	254.0	0.750	0.745
	IBEA	258.0	1.630	1.265
	NSGA-II	254.2	1.522	1.231
$050 \times 10 \times 01$	PMRPEA	794.4	2.759	2.267
	PMRPEA-2	640.0	2.776	2.275
	IBEA	640.1	3.398	2.855
	NSGA-II	767.6	3.421	2.870
$050 \times 20 \times 01$	PMRPEA	1000.7	2.119	1.886
	PMRPEA-2	561.0	2.184	1.923
	IBEA	561.0	3.558	2.995
	NSGA-II	934.9	3.109	2.767
$100 \times 10 \times 01$	PMRPEA	1550.6	2.748	2.252
	PMRPEA-2	1060.0	2.817	2.303
	IBEA	1060.5	4.643	4.028
	NSGA-II	1260.3	3.889	3.367
$100 \times 20 \times 01$	PMRPEA	1740.5	2.666	2.280
	PMRPEA-2	1620.0	2.665	2.285
	IBEA	1620.3	3.424	2.872
	NSGA-II	1702.8	3.277	2.965

TABLE 3.7 – Comparaison des algorithmes selon l'indicateur I_H^-. Pour chaque instance, soit l'algorithme situé sur une ligne donnée domine significativement l'algorithme situé sur une colonne donnée ($\succ$), soit il est significativement dominé ($\prec$), soit il n'existe pas de différence significative entre les deux algorithmes ($\equiv$).

		IBEA	NSGA-II	PMRPEA-2
$020 \times 05 \times 01$	PMRPEA	$\succ$	$\succ$	$\succ$
	PMRPEA-2	$\succ$	$\succ$	-
$020 \times 10 \times 01$	PMRPEA	$\succ$	$\succ$	$\succ$
	PMRPEA-2	$\succ$	$\succ$	-
$020 \times 20 \times 01$	PMRPEA	$\succ$	$\succ$	$\succ$
	PMRPEA-2	$\succ$	$\succ$	-
$050 \times 05 \times 01$	PMRPEA	$\succ$	$\succ$	$\succ$
	PMRPEA-2	$\succ$	$\succ$	-
$050 \times 10 \times 01$	PMRPEA	$\succ$	$\succ$	$\succ$
	PMRPEA-2	$\succ$	$\succ$	-
$050 \times 20 \times 01$	PMRPEA	$\succ$	$\succ$	$\succ$
	PMRPEA-2	$\succ$	$\succ$	-
$100 \times 10 \times 01$	PMRPEA	$\succ$	$\succ$	$\equiv$
	PMRPEA-2	$\succ$	$\succ$	-
$100 \times 20 \times 01$	PMRPEA	$\succ$	$\succ$	$\equiv$
	PMRPEA-2	$\succ$	$\succ$	-

TABLE 3.8 – Comparaison des algorithmes selon l'indicateur $I_{\epsilon+}$. Pour chaque instance, soit l'algorithme situé sur une ligne donnée domine significativement l'algorithme situé sur une colonne donnée ($\succ$), soit il est significativement dominé ($\prec$), soit il n'existe pas de différence significative entre les deux algorithmes ($\equiv$).

		IBEA	NSGA-II	PMRPEA-2
$020 \times 05 \times 01$	PMRPEA	$\succ$	$\succ$	$\equiv$
	PMRPEA-2	$\succ$	$\succ$	-
$020 \times 10 \times 01$	PMRPEA	$\succ$	$\succ$	$\succ$
	PMRPEA-2	$\succ$	$\succ$	-
$020 \times 20 \times 01$	PMRPEA	$\succ$	$\succ$	$\succ$
	PMRPEA-2	$\succ$	$\succ$	-
$050 \times 05 \times 01$	PMRPEA	$\succ$	$\succ$	$\succ$
	PMRPEA-2	$\succ$	$\succ$	-
$050 \times 10 \times 01$	PMRPEA	$\succ$	$\succ$	$\equiv$
	PMRPEA-2	$\succ$	$\succ$	-
$050 \times 20 \times 01$	PMRPEA	$\succ$	$\succ$	$\succ$
	PMRPEA-2	$\succ$	$\succ$	-
$100 \times 10 \times 01$	PMRPEA	$\succ$	$\succ$	$\equiv$
	PMRPEA-2	$\succ$	$\succ$	-
$100 \times 20 \times 01$	PMRPEA	$\succ$	$\succ$	$\equiv$
	PMRPEA-2	$\succ$	$\succ$	-

Conclusion

Ce chapitre nous a permis d'étudier l'hybridation de métaheuristiques dans le cadre de l'optimisation multiobjectif. L'intérêt de ce type d'approche coopérative a été exposé par rapport aux métaheuristiques classiques. Nous avons proposé de nouveaux modèles de coopération entre différentes classes de métaheuristiques, en nous concentrant exclusivement sur des hybridations de haut niveau, qui consistent à combiner un ensemble de métaheuristiques vues comme des agents autonomes. Les principales avancées du chapitre sont récapitulées ci-dessous.

Une approche hybride en mode relais. En premier lieu, nous avons conçu un modèle de coopération général pour la résolution de problèmes d'optimisation multiobjectif. Celui-ci se base sur deux métaheuristiques complémentaires : un algorithme évolutionnaire et un algorithme de recherche locale, dont les choix se sont respectivement porté vers l'algorithme SEEA et l'algorithme IBMOLS lors de l'instanciation du modèle. Nous avons illustré la contribution de ce mécanisme d'hybridation dans le but d'améliorer les performances des méthodes traditionnelles. Il consiste à utiliser l'algorithme évolutionnaire comme processus principal et à lancer la recherche locale, de façon régulière, de sorte à intensifier la recherche. La première variante de ce modèle, la recherche coopérative périodique (PCS), se base sur un déclenchement systématique de la recherche locale à chacune de ses étapes. La seconde variante, la recherche coopérative adaptative (ACS), n'exécute la recherche locale à une certaine étape du processus général que si une condition préalablement définie est vérifiée. En effet, l'approche ACS évolue de façon adaptative vis-à-vis du contexte courant de recherche, et décide par elle-même, en cours de recherche, si la coopération doit se produire. Un autre avantage offert par cette méthode hybride est la possibilité d'orienter délibérément l'équilibre entre la recherche évolutionnaire et la recherche locale. Il pourrait donc se montrer intéressant d'étudier la performance de cette approche à l'égard de différentes spécifications de l'équilibre entre recherches locale et évolutionnaire.
En comparaison avec les métaheuristiques développées lors du premier chapitre pour le problème de Ring-Star, ces deux méthodes hybrides nous ont permis d'améliorer statistiquement les résultats sur un nombre important d'instances, en particulier les instances de grande taille. Cependant, la différence de performance entre les approches PCS et ACS s'est avérée presque négligeable. Cela peut s'expliquer par le fait que la méthode ACS consacre, à chaque étape du processus principal, un temps de calcul non négligeable à déterminer si la coopération doit se produire ou non. Ce n'est pas le cas de l'approche PCS, de sorte que cette dernière consacre la totalité du temps de calcul qui lui est imparti à la recherche de nouvelles solutions.

Une approche hybride en mode teamwork. Dans un second temps, une approche hybride et parallèle, basée sur de multiples points de référence, a été proposée pour résoudre des problèmes d'optimisation multiobjectif. La parallélisation d'algorithmes évolutionnaires a été combinée à l'utilisation d'une approche par point de référence. Le problème peut se formuler comme suit : comment paralléliser une méthode de résolution en divisant l'espace objectif en un certain nombre de régions, puis en appliquant un algorithme évolutionnaire dans chacune de ces régions, et enfin en associant les résultats ? La motivation claire consiste à exploiter la puissance du calcul parallèle. D'autre part, le concept de point de référence peut être naturellement utilisé pour des régions d'intérêt distinctes de l'espace objectif, et donc à la parallélisation des calculs.

L'idée principale consiste donc à diviser uniformément la région couverte par le front Pareto à l'aide de plusieurs points de référence, et à calculer une approximation d'un sous-ensemble de solutions non-dominées associé à chaque point de référence.
Nos expérimentations ont été menées en environnement distribué pour la résolution du problème de Flowshop biobjectif étudié. Les résultats obtenus montrent que l'approche hybride et parallèle proposée donne des résultats très performants, que ce soit en termes de temps de calcul ou en termes de qualité de l'approximation de l'ensemble Pareto optimal. Cependant, de nombreuses questions ouvertes restent à étudier dans de futurs travaux, comme une analyse statistique du facteur d'accélération (« *speed-up* ») dû à la parallélisation, diverses versions possibles de la parallélisation, l'impact d'un plus grand nombre d'objectifs, etc. Nous en discuterons au sein de la conclusion.

L'efficacité des deux approches coopératives proposées au sein de ce chapitre a été démontrée expérimentalement par l'amélioration significative des résultats obtenus par rapport aux métaheuristiques séquentielles et non hybrides équivalentes. Par la suite, nous allons discuter de la résolution de problèmes d'optimisation multiobjectif en environnement incertain.

Chapitre 4

Métaheuristiques pour l'optimisation multiobjectif en environnement incertain

Ce dernier chapitre traite de la résolution de problèmes d'optimisation multiobjectif stochastiques à l'aide de métaheuristiques. Il s'agit d'un domaine de recherche ayant connu un essor indéniable au cours des dernières années, mais pour lequel beaucoup de questions restent encore ouvertes. Les contributions essentielles du chapitre sont les suivantes.

- *Un état de l'art et une classification des approches de résolution pour l'optimisation multiobjectif en environnement incertain.*
- *De nouvelles métaheuristiques basées sur un indicateur de qualité, et une discussion approfondie à propos de l'évaluation des performances (Liefooghe* et al.*, 2009b).*
- *La résolution du problème de Flowshop biobjectif avec durées d'exécutions stochastiques (Liefooghe* et al.*, 2007a).*

Sommaire

Principales publications en rapport avec le chapitre

LIEFOOGHE, A., BASSEUR, M., JOURDAN, L. et TALBI, E.-G. (2007). Combinatorial optimization of stochastic multi-objective problems : an application to the flow-shop scheduling problem. *In Fourth International Conference on Evolutionary Multi-criterion Optimization (EMO 2007)*, volume 4403 de *Lecture Notes in Computer Science*, pages 457–471, Matsushima, Japan. Springer-Verlag.

LIEFOOGHE, A., JOURDAN, L., BASSEUR, M. et TALBI, E.-G. (2008). Métaheuristiques pour le flow-shop de permutation bi-objectif stochastique. *Revue d'Intelligence Artificielle (RIA)*, 22(2):183–208.

Articles soumis

LIEFOOGHE, A., JOURDAN, L. et TALBI, E.-G. (2009). Evolutionary multiobjective optimization in uncertain environments : New approaches, performance assessment and a comparative study on flowshop scheduling. *IEEE Transactions on Evolutionary Computation*. (soumis).

Introduction

À l'heure actuelle, une demande croissante existe pour la résolution de problèmes d'optimisation issus du monde réel, pour lesquels les données ou les fonctions objectif sont souvent incertaines, et où il s'avère parfois nécessaire de trouver des solutions robustes à des modifications ultérieures susceptibles de survenir sur les variables de décision. En pratique, résoudre ce type de problème se résume souvent à une transformation du problème original, soumis à incertitude, en un problème déterministe. Pourtant, certaines adaptations en termes de méthodes de résolution peuvent s'avérer utiles. Ce domaine de recherche a reçu un intérêt grandissant ces dernières années en raison de sa difficulté et de son importance en pratique. De par leur nature stochastique intrinsèque, les métaheuristiques conventionnelles, adaptées de façon adéquate, présentent des mécanismes intéressants pour la résolution de problèmes d'optimisation sous incertitude. Mais, bien qu'elles soient couramment utilisées pour résoudre des problèmes multiobjectif d'une part, et stochastiques d'autre part, très peu d'études ont été menées de front sur ces deux thématiques. À titre d'exemple, les problèmes d'ordonnancement sont habituellement traités sous une forme monoobjectif déterministe. Cependant, ils sont bien souvent manifestement multiobjectif (T'Kindt et Billaut, 2002) et ils sont soumis à de nombreux facteurs d'incertitude (Billaut *et al.*, 2005). Or, malgré l'importance de l'optimisation multiobjectif, et de l'optimisation en environnement incertain, un nombre très restreint de travaux consacrés à la prise en compte de ces deux aspects existe à ce jour, et se limite généralement à des problèmes multiobjectif continus.

Le présent chapitre se veut délibérément plus exploratoire que les chapitres précédents. Nous nous efforçons ici d'aborder les questions essentielles à se poser lors de la prise en compte de l'incertitude dans un contexte multiobjectif, et lors de la résolution de problèmes d'optimisation multiobjectif en environnement incertain. Nous mettons en avant que le concept de base de l'incertitude se traduit, dans bien des cas, par l'application de tels ensembles depuis l'espace décisionnel vers l'espace objectif. La prise en compte de l'incertitude entraîne donc la comparaison d'ensembles. D'un point de vue multiobjectif, les solutions sont projetées vers des ensembles (ou des échantillons) de vecteurs objectif dont la forme n'est pas nécessairement connue *a priori* par le praticien ou par le décideur. Le véritable challenge soulevé par ce type de problème peut donc se résumer à la comparaison deux à deux d'ensembles de vecteurs objectif, plutôt que de vecteurs objectif uniques dans le cas déterministe. Les méthodes de résolution ainsi que les protocoles d'évaluation des performances doivent donc être adaptés afin de prendre cette spécificité en compte, que ce soit par le biais d'une sélection de vecteurs représentatifs ou par l'adaptation de leurs mécanismes internes.

Dans ce chapitre, nous fournissons d'abord un panorama qui se veut exhaustif et représentatif de l'état de l'art actuel en termes de métaheuristiques pour l'optimisation multiobjectif en environnement incertain (section 4.1). Nous proposons également une classification des différentes approches de résolution. Dans la section 4.2, nous présentons un ensemble de nouvelles approches dédiées à la résolution de ce type de problème. Il s'agit de métaheuristiques basées sur un indicateur de qualité qui adaptent donc l'approche proposée par Zitzler et Künzli (2004) à la prise en compte de l'incertitude, que ce soit sur les fonctions objectif, les paramètres environnementaux ou les variables de décision. La section 4.4 discute de l'analyse de la performance des méthodes de résolution dans un tel cas. Puis, nous introduisons plusieurs modèles de représentation de l'incertitude pour le problème de Flowshop. Les diverses propositions du chapitre sont ensuite mises en pratique et expérimentées sur le problème de Flowshop pour lequel les durées d'exécu-

tions sont dorénavant considérées comme incertaines. À notre connaissance, c'est la première fois qu'un problème d'ordonnancement stochastique est étudié sous une formulation multiobjectif.

4.1 Classification des métaheuristiques pour l'optimisation multiobjectif en environnement incertain

Dans cette section, nous présentons, dans un premier temps, les classes d'incertitude susceptibles d'intervenir dans la formulation d'un problème d'optimisation stochastique. Puis, nous interprétons ces différentes classes pour le cas multiobjectif. Pour finir, une classification des approches de résolution est proposée pour ce type de problème.

4.1.1 Contexte

4.1.1.1 Classes d'incertitude

Dans leur étude sur les approches d'optimisation évolutionnaires en environnement incertain, Jin et Branke (2005) classent les différents types d'incertitude en quatre catégories, de la façon suivante.

1. **Fonction objectif bruitée**. L'évaluation d'une solution est sujette au bruit. Ce bruit peut provenir de différentes sources (erreur de mesure sensorielle, simulations aléatoires). Pour ce genre de problème, à chaque appel de la fonction objectif avec les mêmes arguments, celle-ci retourne une valeur différente.
2. **Recherche de solutions robustes**. Ici, les variables de conception sont susceptibles d'être soumises à des perturbations ou à des modifications à la suite du processus de recherche. Par conséquent, il est fréquemment exigé qu'une solution reste satisfaisante suite à une légère variation des variables de décision, en raison par exemple de tolérances de fabrication. De telles solutions sont dites *robustes*. Dans l'idéal, une solution robuste doit donc être insensible à de petites variations sur les variables de décision. Dans un tel cas, la fonction objectif est généralement connue et déterministe, l'incertitude étant uniquement introduite après le processus de recherche. Par ailleurs, comme soulevé par les auteurs, il se peut que des perturbations apparaissent sur d'autres paramètres que les variables de décision, comme les paramètres environnementaux. Ces deux types d'incertitude sont classés dans une seule et même catégorie par Jin et Branke (2005). Nous pensons plutôt qu'elles résultent en des sources résolument différentes. La prise en compte de paramètres environnementaux incertains est finalement assez proche du cas où la fonction objectif est bruitée, puisque l'incertitude prise en compte ici a une répercussion directe sur le calcul de la fonction objectif. Ainsi, la recherche de solutions robustes peut être traitée à partir de ces deux perspectives. Une solution robuste peut donc être également vue comme insensible à de petites variations sur les paramètres environnementaux.
3. **Fonction objectif approchée**. Lorsque l'évaluation d'une solution est extrêmement coûteuse en temps de calcul, ou qu'une fonction objectif analytique n'est pas disponible, la fonction objectif utilisée est souvent approchée à l'aide de données générées de façon expérimentale ou issues de simulations. En conséquence, l'algorithme doit généralement être en mesure de manipuler deux fonctions : la fonction originale, coûteuse mais exacte, et une fonction de substitution, économique mais imprécise.

4. **Environnement dynamique**. La fonction objectif est ici déterministe à n'importe quel instant, mais est dépendante du temps. En conséquence, l'optimum change également au cours de l'exécution. La méthode mise en place devrait donc être en mesure de pouvoir continuellement suivre le changement de l'optimum plutôt que de nécessiter un redémarrage complet du processus de recherche.

Les deux dernières classes présentées ci-dessus ne sont pas abordées au sein de cette thèse. Premièrement, nous pensons que la considération d'une fonction objectif approchée est étroitement liée au problème étudié. Or, nous essayons ici d'être aussi généraux que possible d'un point de vue méthodologique. Deuxièmement, nous pensons que la prise en compte d'une fonction objectif dynamique au sein d'une approche réactive est une question totalement différente que de manipuler les autres types d'incertitude. Faire face à une fonction objectif dynamique à l'aide d'une approche proactive revient, selon les particularités du problème, à la prise en compte d'une fonction objectif bruitée, ou à la recherche de solutions robustes vis-à-vis de paramètres environnementaux incertains. Nous nous concentrerons donc sur les deux premières classes d'incertitude : la prise en compte d'une fonction objectif bruitée et la recherche de solutions robustes à des perturbations sur les variables de décision ou sur les paramètres environnementaux.
En pratique, lors du premier cas, la fonction objectif bruitée est souvent estimée à l'aide d'une valeur moyenne explicitement ou implicitement calculée sur un échantillon de valeurs objectif. Lors de la recherche de solutions robustes, la fonction objectif est généralement approchée à l'aide d'une intégration de Monte-Carlo. Ainsi, cette approximation est très proche de celle couramment utilisée en présence de bruit, de sorte que ces deux classes sont étroitement liées l'une à l'autre. En effet, dans tous les cas, la méthodologie la plus classique consiste à établir un échantillon de valeurs objectif, ou de vecteurs de décision ultérieurement transformés en valeurs objectif. Ainsi, peu importe la façon dont l'incertitude est propagée, les résultats deviennent des ensembles de valeurs, et donc des ensembles de vecteurs objectif dans le cadre de l'optimisation multiobjectif, comme nous le verrons par la suite. En conséquence, la difficulté soulevée concerne la définition d'optimalité en termes d'ensembles.

Jin et Branke (2005) remarquent que très peu de travaux sont consacrés à la résolution et au traitement de problèmes d'optimisation multiobjectif en environnement incertain. Ainsi, la première classe d'incertitude, pour laquelle les fonctions objectif sont bruitées, ont été indépendamment considérées par Teich (2001) et Hughes (2001). Par ailleurs, la robustesse de solutions Pareto optimales pour des problèmes d'optimisation multiobjectif où des variables de décision continues sont incertaines a fait l'objet d'une étude approfondie par Deb et Gupta (2006). Nous reviendrons sur ces travaux en détail par la suite.
Cependant, certaines approches issues de l'optimisation multiobjectif sont parfois considérées en vue de la résolution de problèmes monoobjectif incertains. Dans un tel cas, une fonction objectif proche de l'originale, ainsi qu'un critère de robustesse, sont tous deux considérés simultanément comme des objectifs distincts. Le but est ici d'obtenir de multiples compromis entre performance et robustesse. Par exemple, Jin et Sendhoff (2003) proposent de considérer à la fois l'espérance mathématique et une mesure de variance d'un ensemble de valeurs observées sur la fonction objectif bruitée originale, obtenant ainsi deux fonctions objectif symbolisant respectivement la qualité et la robustesse d'une solution.

À propos de robustesse. La notion de *robustesse* étant souvent sujette à discussion, il nous a semblé utile de nous y attarder plus longuement. En effet, il n'existe pas de définition unanime

de la robustesse dans la littérature. En règle générale, le concept de robustesse se réfère à la capacité du sujet à faire face à des incertitudes. Roy (2005) définit le terme robuste comme *un qualificatif se rapportant à une aptitude à résister à des « à peu près » ou à des « zones d'ignorances » afin de se protéger d'impacts jugés regrettables* (Roy, 2005). En effet, l'auteur remarque que la notion de robustesse ne doit pas être uniquement appliquée aux solutions, mais plus généralement à différentes affirmations et recommandations générées au sein d'un système informatique d'aide à la décision. Tout d'abord, nous allons distinguer une *solution robuste* et une *méthode de résolution robuste*, cette dernière se rapportant généralement à son extensibilité. Cependant, une vue commune est partagée : une solution robuste doit être en mesure de bien se comporter sous des conditions (légèrement) différentes, ce qui signifie qu'elle doit, autant que possible, être à l'abri de petites modifications au sein des conditions dans lesquelles elle a été conçue. De façon générale, la prise en compte de la robustesse d'une solution se doit de distinguer les variables de décisions déterministes, les variables de décisions stochastiques, et les paramètres extérieurs.
Ainsi, nous allons considérer qu'une solution à un problème d'optimisation stochastique est robuste si elle conserve une haute performance indépendamment d'une certaine réalisation des données incertaines. Cette notion est donc à la fois liée à la qualité d'une solution ainsi qu'à sa variabilité, ou sa stabilité. Par ailleurs, la robustesse d'une solution se rapporte habituellement à la fiabilité et à la flexibilité de cette même solution. La *fiabilité* d'une solution est liée à la probabilité que celle-ci reste réalisable en présence d'incertitude, même s'il est en général difficile de totalement exclure un tel risque. Par ailleurs, il peut s'avérer possible d'ajuster une solution en fonction de tels événements. La *flexibilité* d'une solution peut être définie comme sa capacité à pouvoir être adaptée à la suite d'une réalisation des données incertaines.

4.1.1.2 Incertitude en optimisation multiobjectif

Une approche de résolution très courante en optimisation monoobjectif sous incertitude (Jin et Branke, 2005) consiste donc à établir un échantillon de valeurs objectif associé à chaque solution réalisable. Néanmoins, nous pouvons distinguer le cas où une fonction objectif est intrinsèquement incertaine (présence de bruit, environnement dynamique) du cas où la fonction objectif est une expression fonctionnelle déterministe (robustesse). Lors du passage à l'optimisation multiobjectif, la présence d'incertitude entraîne la comparaison d'ensembles de vecteurs objectif à la place d'un vecteur unique et précis. Ci-dessous, nous tentons d'expliquer la façon méthodologique de procéder afin de générer un échantillon de vecteurs objectif associé à une solution dans le cadre de l'optimisation multiobjectif, ceci pour chacune des classes d'incertitude étudiées.

- **Fonctions objectif bruitées**. Dans ce cas, les fonctions objectif sont directement sujettes au bruit, la seule évaluation disponible est donc intrinsèquement stochastique. Ainsi, soit une distribution de probabilité connue ou présumée est considérée, ce qui semble peu probable en pratique, soit il s'avère nécessaire d'évaluer à plusieurs reprises, et de façon indépendante, une même solution en vue d'obtenir un ensemble fini de résultats possibles (Fig. 4.1). C'est cette dernière alternative que nous allons considérer par la suite. De ce fait, aucune évaluation ne peut être considérée plus probable qu'une autre, et encore moins comme évaluation réelle, ou comme évaluation de référence.
- **Robustesse vis-à-vis des paramètres environnementaux**. Ici, les fonctions objectif ne sont pas intrinsèquement stochastiques. Néanmoins, lors de l'évaluation d'une solution, celles-ci requièrent un certain nombre de paramètres qui peuvent être aléatoires. Pour tout paramètre

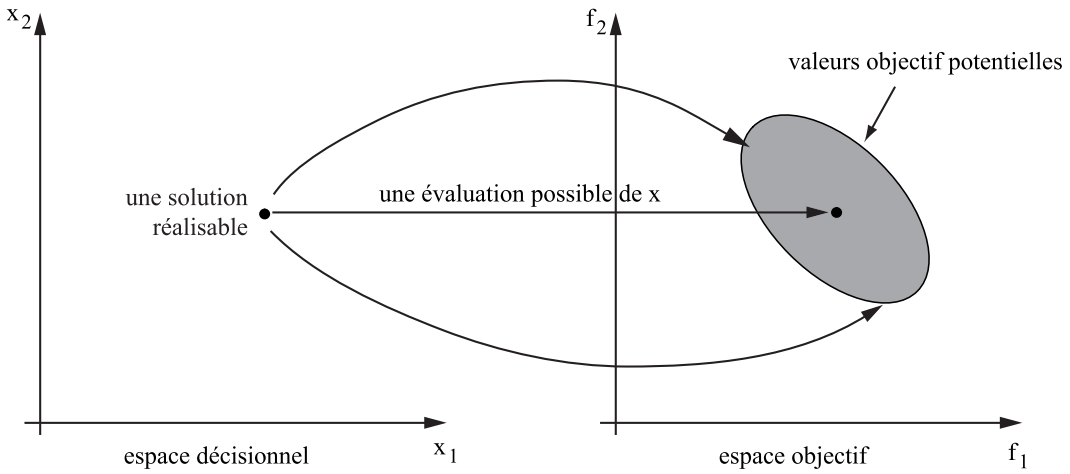


Fig. 4.1 – Prise en compte de l'incertitude à l'aide d'un échantillon de vecteurs objectif pour les fonctions objectifs bruitées et pour la recherche de solutions robustes vis-à-vis de paramètres environnementaux incertains.

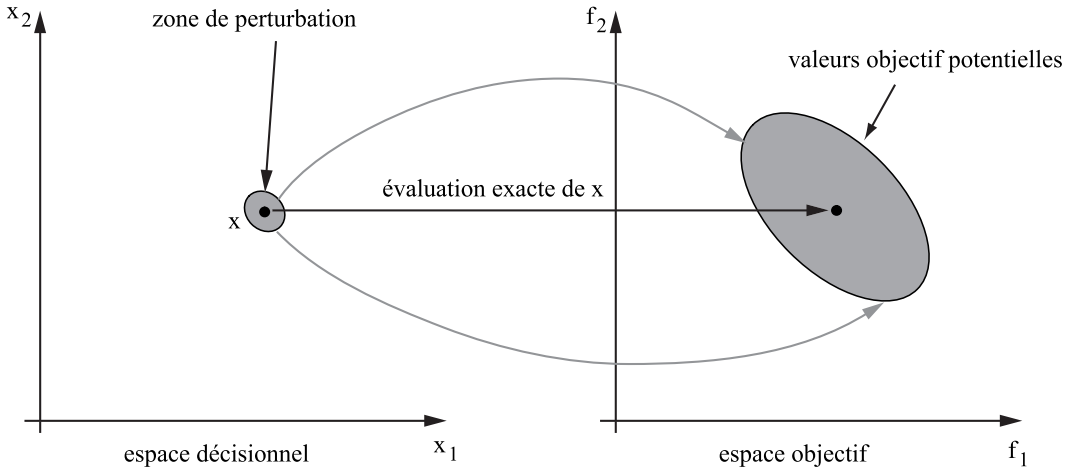


Fig. 4.2 – Prise en compte de l'incertitude à l'aide d'un échantillon de vecteurs objectif pour la recherche de solutions robustes vis-à-vis de variables de décision incertaines.

environnemental incertain, un ensemble de valeurs possibles est considéré, chaque combinaison donnant lieu à une évaluation potentiellement différente pour une même solution. Ainsi, de la même manière que pour le cas précédent, nous allons comptabiliser un ensemble fini de vecteurs objectif possibles (Fig. 4.1). La prise en compte de ce type d'incertitude est donc relativement semblable au cas précédent.

– **Robustesse vis-à-vis des variables de décision**. Ce dernier type d'incertitude se montre légèrement différent des deux autres. En effet, à chaque solution réalisable correspond bien un vecteur objectif unique. Autrement dit, l'évaluation réelle d'une solution peut être calculée. Néanmoins, les variables de décision sont ici susceptibles d'être modifiées à la suite du processus de recherche. Ainsi, la solution trouvée se doit d'être satisfaisante, et donc insensible (dans la mesure du possible) à la perturbation des variables de décision dans son voisinage. En pratique, un ensemble fini de solutions peut être défini (aléatoirement ou de façon structurée) autour d'une solution, dans l'espace décisionnel ; ceci selon un opérateur de voisinage donné. Une des difficultés réside donc en la définition d'un opérateur de voisinage pertinent pour le problème sous incertitude à résoudre. À la place d'une seule évaluation par solution, un ensemble fini de vecteurs objectif, correspondant à l'évaluation des solutions voisines, est encore une fois considéré ; même s'il est vrai que l'évaluation réelle et exacte d'une solution donnée est ici connue (Fig. 4.2). L'intention est donc de tenir compte de la sensibilité des valeurs objectif dues à l'incertitude des variables de décision dans un contexte multiobjectif.

Il se peut que certains problèmes soient sujets à plusieurs types d'incertitude simultanément, auquel cas un ensemble d'ensembles de vecteurs objectif est dès lors associé à chaque solution réalisable.

Comme nous venons de le voir, dès que la prise en compte de l'incertitude se base sur un échantillonnage, cela entraîne la comparaison d'ensembles de vecteurs objectif plutôt que vecteurs objectif uniques. Par conséquent, la question fondamentale, et donc le véritable challenge soulevé par ce type de problème, se résume à pouvoir comparer et discriminer différentes solutions alternatives en termes de comparaison d'ensembles de vecteurs objectif (Fig. 4.3). Ainsi, cet aspect donne lieu à la principale différence entre de la conception de métaheuristiques pour l'optimisation de problèmes multiobjectif déterministes et stochastiques.

4.1.1.3 Approches issues de la programmation mathématique

D'un point de vue historique, plusieurs classes de méthodologies sont dédiées à la résolution de problèmes d'optimisation stochastiques. Tout d'abord, il existe un champ de recherche connu sous le nom d'optimisation robuste (*robust optimization*) (Kouvelis et Yu, 1997). L'optimisation robuste est une approche proactive qui traite de l'incertitude des données d'un problème d'optimisation. Ces données incertaines sont représentées à l'aide d'un ensemble discret de différents scénarios introduits au sein d'un problème d'optimisation pour se prémunir de l'incertitude ; un scénario correspondant en quelque sorte à une réalisation des paramètres environnementaux. En général, les approches classiques se basent sur le scénario reflétant le *pire des cas* pour chaque solution (Kouvelis et Yu, 1997). Ce type d'approche présente donc une haute aversion au risque. La programmation stochastique multiobjectif (*multiobjective stochastic programming*) est un domaine de la programmation mathématique classique qui tient compte de l'incertitude et de l'aspect multiobjectif du problème (Slowinski et Teghem, 1990). C'est en quelque sorte la fusion des domaines de la programmation stochastique et de la programmation multiobjectif. En règle générale, la programmation stochastique se base sur des distributions de probabilité associées

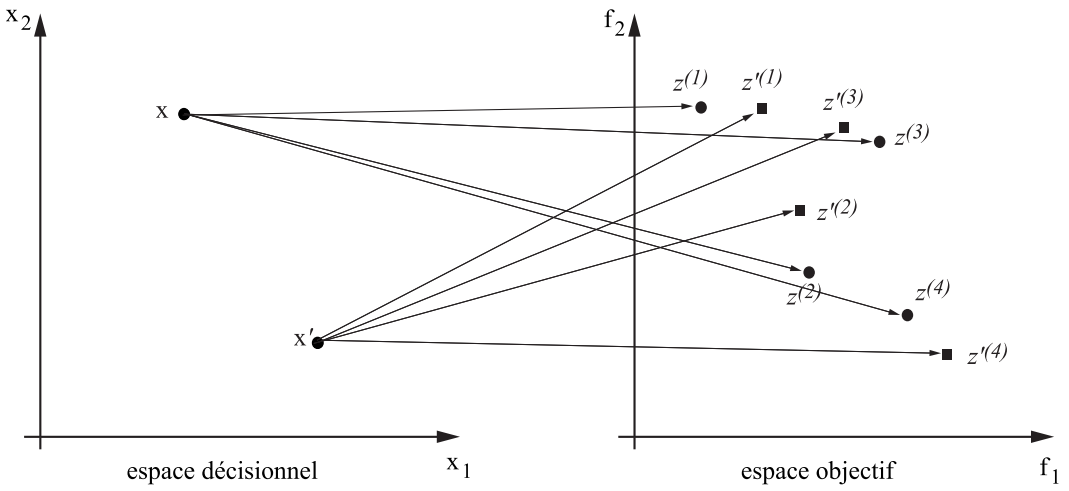


Fig. 4.3 – Illustration de la comparaison de deux ensembles de vecteurs objectif $\{z^{(i)}\}_{i=1}^{4}$ et $\{z'^{(j)}\}_{j=1}^{4}$ obtenus de l'évaluation multiple de deux solutions $x, x' \in X$. Quelle solution est meilleure que l'autre ?

aux variables de décision ou aux paramètres environnementaux, et sur l'optimisation de l'espérance mathématique ; ce type d'approche correspond donc plutôt au *cas moyen*. Caballero *et al.* (2004) distinguent deux types d'approche ; elles sont illustrées sur la figure 4.4. Premièrement, les *approches multiobjectif* consistent à (*i*) réduire le problème multiobjectif stochastique initial en un problème multiobjectif déterministe, (*ii*) réduire le problème multiobjectif déterministe en un problème monoobjectif déterministe. Deuxièmement, les *approches stochastiques* opèrent ces deux mêmes transformations dans l'ordre inverse.

Quant à elle, l'optimisation multiobjectif floue (*fuzzy multiobjective optimisation*) correspond plutôt à des techniques d'optimisation multiobjectif où l'incertitude se situe au niveau des préférences du décideur, formalisées à l'aide de la logique floue, et non à l'incertitude des données ou de la fonction objectif.

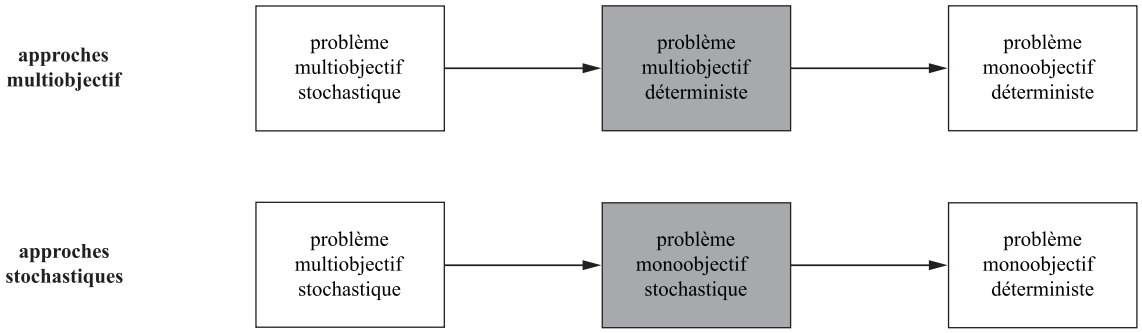


Fig. 4.4 – Deux types d'approche au sein de la programmation stochastique multiobjectif : les approches multiobjectif et les approches stochastiques (Caballero *et al.*, 2004).

4.1.2 Approches métaheuristiques

Dans cette section, nous essayons de fournir un aperçu global des méthodes de résolution en termes de métaheuristiques pour les problèmes d'optimisation multiobjectif sous incertitude. Une première tentative a déjà été faite en ce sens par Tan et Goh (2008). Néanmoins, cette dernière donne plutôt une vue d'ensemble de la contribution des auteurs sur ce thème de recherche, se concentre uniquement sur les algorithmes évolutionnaires, et traite des divers types d'incertitude depuis des perspectives différentes en terme algorithmique. Ici, nous essayons de fournir une vue commune pour tous les types d'incertitude (présence de bruit, robustesse vis-à-vis des variables de décision, robustesse vis-à-vis des paramètres environnementaux, et résolution proactive de problèmes dynamiques), ainsi qu'une classification générale des stratégies existantes se basant sur un échantillonnage des fonctions objectif.

Comme nous l'avons déjà remarqué, l'un des défis majeurs soulevés par l'optimisation de problèmes multiobjectif et stochastiques est de comparer et classer différentes alternatives en termes d'ensembles de comparaisons, c'est-à-dire d'échantillons de vecteurs objectif. La classification que nous proposons par la suite est divisée en deux niveaux. Tout d'abord, il se peut qu'il existe une phase initiale consistant à choisir, pour chaque solution évaluée, un ou plusieurs vecteurs représentatifs ; ou plutôt une ou plusieurs valeurs représentatives pour chacune des fonctions objectif. Cet ensemble de valeurs peut tout à fait être constitué de l'échantillon complet d'informations dont nous disposons, *a priori*, pour une solution donnée. Bien sûr, ce dernier échantillon est également apparenté au nombre d'évaluations différentes qui a été décidé pour la solution donnée. Ensuite, nous passerons en revue les différentes méthodes qui ont été proposées jusqu'à présent pour tenir compte de ces valeurs représentatives au sein d'approches métaheuristiques. Évidemment, ces approches sont bien souvent liées à la décision des valeurs représentatives considérées. Une classification de ces différents types de méthode est proposée, ceci dans le but de fournir une dénomination commune à un ensemble de solutions algorithmiques (Fig. 4.5). Bien que la quasi-totalité des méthodes présentées se base sur les algorithmes évolutionnaires, il nous a semblé plus pertinent de les présenter en dehors de tout cadre métaheuristique car elles ne dépendent généralement pas de ce formalisme méthodologique. Les choses sont donc présentées de façon incrémentale, et se basent sur les modèles de conception présentés au cours des chapitres précédents, en particulier le chapitre 2.

4.1.2.1 Choix de valeurs représentatives

La plupart des techniques visant à déterminer des valeurs représentatives d'un ensemble d'informations sont des approches scalaires, qui consistent à agréger un échantillon de données en une valeur scalaire unique, et qui résultent donc en la conversion de la solution imprécise en un vecteur objectif unique. La comparaison des ensembles est alors réduite à une comparaison de points représentatifs. Ainsi, la formulation résultante du problème devient déterministe et une méthode de résolution usuelle peut être envisagée. Une telle agrégation peut, par exemple, porter sur l'échantillon de valeurs objectif obtenu par une solution, ceci pour chacune des fonctions objectif considérées. Mais elle peut aussi porter sur les variables ou sur les paramètres stochastiques intervenant dans la formulation du problème.
Néanmoins, ne considérer qu'une seule valeur représentative par objectif, telle que la valeur moyenne, la valeur dans le pire des cas ou n'importe quelle autre valeur scalaire qui dénote la quantité d'incertitude, ne s'avère pas toujours fiable et provoque généralement une haute

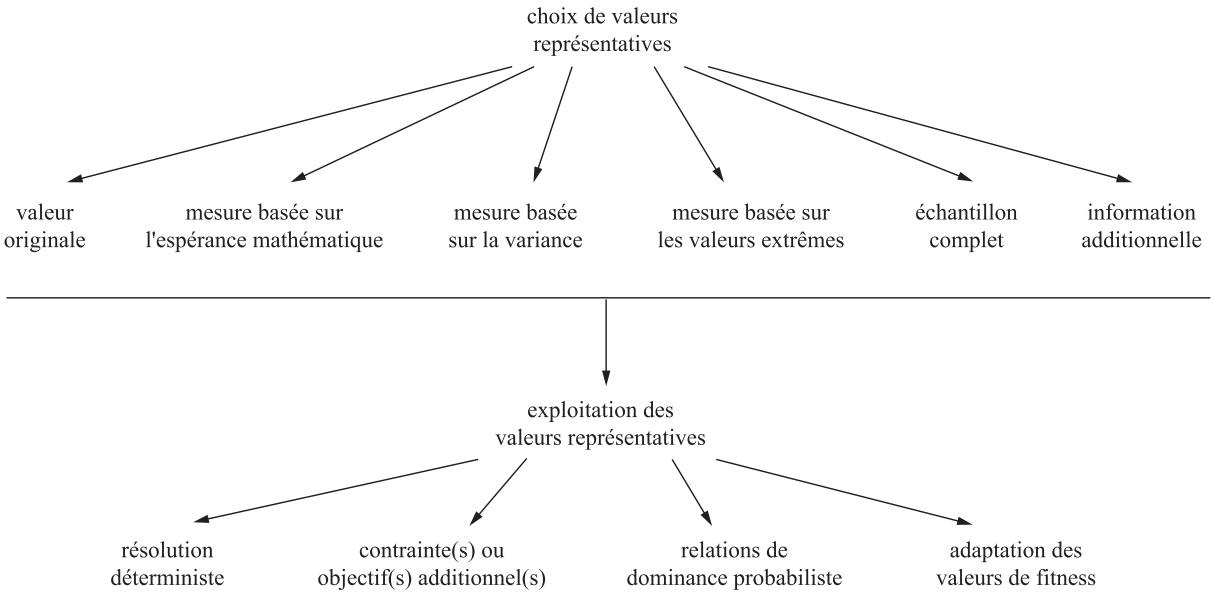


Fig. 4.5 – Classification des approches de résolution.

perte d'information. Typiquement, beaucoup de stratégies de résolution estiment la robustesse d'une solution à l'aide de la moyenne et de la variance, qui sont alors utilisées comme valeurs représentatives. Ci-dessous, nous présentons différents types de valeur représentative susceptible d'être considérée à partir de l'échantillon de valeurs obtenu par une solution pour une fonction objectif donnée. Par la suite, nous allons détailler différentes approches de résolution visant à exploiter une valeur représentative, ou un ensemble de valeurs représentatives, dans le cadre de l'optimisation multiobjectif.

4.1.2.1.1 Valeur originale. Tout d'abord, si la prise en compte de l'incertitude pour le problème à résoudre vise à identifier des solutions robustes vis-à-vis des variables de décision, l'évaluation réelle d'une solution peut être calculée, et peut donc servir comme valeur représentative pour chaque fonction objectif (Deb et Gupta, 2006; Goh et Tan, 2007a). Ce type d'incertitude est le seul pour lequel une telle information est disponible. Toutefois, si seule cette valeur originale est considérée au sein de l'ensemble des valeurs jugées représentatives, la méthode de résolution se comportera de la même manière que pour un problème déterministe, et l'incertitude ne sera pas prise en compte. C'est la raison pour laquelle la valeur originale unique ne peut pas être considérée de façon exclusive, et doit donc être complétée d'un ou plusieurs autres éléments d'information.

4.1.2.1.2 Mesures basées sur l'espérance mathématique. Une forme typique de scalarisation consiste à approcher l'espérance mathématique de la, ou des données incertaines à l'aide de la valeur moyenne observée sur un échantillon de valeurs, même si la valeur médiane peut également être utilisée. Dans le cas d'une scalarisation des valeurs observées sur un objectif, une telle stratégie prend en compte la valeur moyenne de p évaluations d'une même solution, ce qui permet de générer des solutions de bonne qualité, en moyenne. Dans le cadre de l'opti-

misation multiobjectif, c'est généralement le vecteur objectif moyen qui est considéré de façon explicite (Bui *et al.*, 2005; Deb et Gupta, 2006; Tan *et al.*, 2007; Gaspar-Cunha et Covas, 2008). À chaque solution est affecté un échantillon de vecteurs objectif, et la valeur moyenne par fonction objectif est calculée pour représenter une composante du vecteur objectif moyen. Ainsi, on peut raisonnablement penser qu'agrandir la taille de l'échantillon a pour effet de proportionnellement réduire le degré d'incertitude, au détriment d'un coût de calcul plus élevé.

4.1.2.1.3 Mesures basées sur la variance. Comme nous l'avons déjà remarqué, une approche naturelle en vue de mesurer la robustesse d'une solution pour un objectif consiste à prendre en compte une estimation de l'espérance mathématique des valeurs objectif, et parfois une mesure de variance additionnelle. Ce type de modèle basé sur la moyenne et la variance de valeurs objectif semble particulièrement bien adapté aux distributions normales. Une mesure de variance, typiquement l'écart-type empirique corrigé, vise à évaluer la dispersion d'un échantillon par rapport à la valeur moyenne. Elle ne prend en compte que les déviations de la fonction, et ignore généralement le critère de performance. Elle peut donc être considérée comme une mesure de l'incertitude d'une variable aléatoire. D'autres mesures d'incertitude, comme l'étendue, l'écart interquartile ou l'entropie, peuvent également être définies.
Néanmoins, considérer à la fois la moyenne (ou la valeur originale) et une mesure d'incertitude par fonction objectif a pour effet de multiplier par deux le nombre de valeurs représentatives par rapport au problème déterministe initial (Jin et Sendhoff, 2003). Pour éviter cela, il faut trouver le moyen de remplacer la mesure de variance de tous les objectifs par une mesure globale unique. Par exemple, Gaspar-Cunha et Covas (2008) proposent de considérer la moyenne des variances observées sur chaque objectif, ou encore la mesure de variance dans le pire des cas. Des approches alternatives existent afin de ne considérer qu'une seule valeur représentative par solution et par fonction objectif. En effet, au sein d'une des stratégies proposées par Bui *et al.* (2005), la valeur moyenne m_i de l'échantillon d'une solution pour la fonction objectif f_i est pénalisée par une mesure de variance σ_i (estimation de l'écart type) observée sur le même objectif (m_i/σ_i).

4.1.2.1.4 Mesures basées sur les valeurs extrêmes. Le pire et le meilleur des cas correspondent aux valeurs extrêmes de l'intervalle de valeurs obtenues par une solution par rapport à une fonction objectif. En effet, dans le cas où la valeur représentative unique est une valeur moyenne, la méthode de résolution cherche à trouver un ensemble de solutions de bonne qualité, en moyenne. Or, en analogie avec le critère de robustesse absolu de Kouvelis et Yu (1997), la prise en compte de la performance d'une solution dans le pire des cas peut tout aussi bien être envisagée, ce qui traduit une haute aversion au risque. Une telle mesure peut être considérée de façon absolue, mais également en addition à une autre valeur représentative (Goh et Tan, 2007a).

Par ailleurs, plutôt que de comparer les solutions deux à deux sur la base d'un seul vecteur dont les éléments proviennent de données agrégées, la prise en compte de l'incertitude sur les n fonctions objectif peut encore résulter en n intervalles de valeurs. Dans ce type d'approche, les deux valeurs extrêmes (minimum et maximum) par fonction objectif sont à la fois prises en compte pour une solution donnée (Teich, 2001; Limbourg, 2005; Limbourg et Salazar Aponte, 2005). Ainsi, les valeurs objectif d'une solution sont des données imprécises représentées à l'aide d'intervalles multidimensionnels : $f(x) = z = ([\underline{z_1}, \overline{z_1}], [\underline{z_2}, \overline{z_2}], \ldots, [\underline{z_n}, \overline{z_n}])$. Le challenge réside donc en la comparaison deux à deux de n éléments bidimensionnels, plutôt que n éléments

unidimensionnels dans le cas déterministe ; auquel cas les méthodes de résolution exploitent ces valeurs par le biais de l'arithmétique d'intervalles.

4.1.2.1.5 Échantillon complet. Il se peut que certaines méthodes de résolution tentent de prendre en compte la totalité de l'échantillon de vecteurs objectif associé à chaque solution, sans déterminer de valeur plus représentative qu'une autre pour chaque fonction objectif. Le choix d'une quelconque technique scalaire est donc situé à un autre niveau que sur les échantillons de vecteurs objectif, et est généralement inhérent à une stratégie d'affectation des valeurs de fitness, ou de diversité, particulière. L'exploitation des valeurs se base sur la comparaison deux à deux d'éléments des échantillons complets ; auquel cas la cardinalité des ensembles présente ici la limitation majeure. Plusieurs exemples de métaheuristiques où l'échantillon complet est considéré seront proposés et présentés dans la section suivante.

4.1.2.1.6 Information additionnelle. Un grand nombre de travaux liés à la résolution de problèmes d'optimisation multiobjectif en environnement incertain supposent l'existence explicite d'une loi de probabilité associée sur l'espace objectif à chaque solution réalisable. Par exemple, le traitement proposé par Hughes (2001) fait l'hypothèse d'une loi normale dont les paramètres (la moyenne et la variance) sont connus, ou dont seule la variance est disponible. L'auteur soutient que d'autres densités de probabilité peuvent être traitées, mais cette approche requiert clairement la connaissance et la mise à disposition d'une distribution de probabilité donnée. De même, Teich (2001) fait l'hypothèse que les variables aléatoires sont uniformément distribuées entre deux valeurs.

D'autres types de mesure représentative peuvent tout aussi bien être envisagés, comme par exemple une mesure dédiée au maintien de la faisabilité d'une solution en environnement incertain, et donc étroitement liée à la fiabilité d'une solution. On pourrait aussi imaginer des mesures de robustesse plus proches, voire spécifiques au problème stochastique étudié. Enfin, à notre connaissance, il n'existe pas d'approche basée sur la logique floue, bien que l'usage de ce type de théorie de l'incertitude semble tout à fait envisageable en pratique.

4.1.2.2 Exploitation des valeurs représentatives

Ci-dessous sont présentées différentes stratégies permettant d'exploiter l'information contenue au sein des valeurs représentatives désignées pour chaque solution. Nous pensons que ces différentes approches sont caractéristiques de l'état de l'art actuel pour la résolution de problèmes d'optimisation multiobjectif stochastiques en termes de métaheuristiques.

4.1.2.2.1 Résolution déterministe. Si l'étape de choix de valeurs représentatives par solution se résume à la sélection d'une valeur scalaire unique par fonction objectif, le problème à résoudre devient un problème d'optimisation multiobjectif déterministe, pour lequel chaque solution se voit affecter n valeurs relatives aux n fonctions objectif. Dans ce cas, une méthode classique peut directement être utilisée afin de résoudre le problème étudié (Deb et Gupta, 2006; Tan *et al.*, 2007). En dehors du choix de la valeur représentative par fonction objectif, aucun traitement particulier quant à la prise en compte de l'incertitude n'est donc accompli.

4.1.2.2.2 Contrainte(s) ou objectif(s) additionnel(s). Contrairement au cas précédent, plus d'une valeur représentative est maintenant disponible par fonction objectif. Ici, un ensemble de contraintes, ou encore un ensemble de fonctions objectif est ajouté ; ce qui a donc pour effet d'augmenter le nombre de contraintes ou d'objectifs par rapport au problème initial déterministe. Cette classe d'approches semble surtout populaire afin de traiter de la robustesse vis-à-vis des variables de décision, où les n fonctions objectif originales sont bien souvent employées telles quelles au sein de l'approche considérée. En effet, ce type d'incertitude correspond au seul cas où l'évaluation réelle d'une solution est disponible.
Par exemple, afin de résoudre un problème multiobjectif pour lequel l'incertitude se trouve sur les variables de décision, Deb et Gupta (2006) proposent de considérer les fonctions objectif du problème déterministe original et d'y ajouter une contrainte sur la déviation maximale par rapport au vecteur objectif original. Ceci permet de contrôler la dispersion à l'aide d'un paramètre de limitation fourni par l'utilisateur. Cependant, cette approche ne permet pas de contrôler la variabilité d'une solution, bien que des travaux complémentaires traitant de problèmes monoobjectif proposent de considérer le pire des cas, ou encore de pondérer l'importance des scénarios (Sörensen, 2003).
Enfin, comme nous l'avons déjà remarqué, la considération d'une fonction objectif additionnelle est une technique relativement courante lors de la résolution d'un problème d'optimisation monoobjectif sous incertitude (Ray, 2002; Jin et Sendhoff, 2003). Elle consiste généralement à considérer simultanément, et comme objectifs distincts, le vecteur objectif moyen, remplacé par le vecteur objectif déterministe dans le cas de la recherche de solutions robustes, voire la considération des deux (Ray, 2002), ainsi qu'une mesure de robustesse. Cette mesure de robustesse peut par exemple concerner la variance (Gaspar-Cunha et Covas, 2008) ou encore la valeur dans le pire des cas (Goh et Tan, 2007a). Des techniques similaires existent également pour le cas où la formulation déterministe du problème stochastique à résoudre est intrinsèquement multiobjectif. Une stratégie immédiate consiste à considérer deux critères (moyenne et robustesse) par fonction objectif originale. Mais ceci a pour effet de multiplier le nombre de critères par deux. Pour éviter cela, il faut trouver le moyen de développer une mesure de robustesse sans augmenter substantiellement la dimension du problème. Ainsi, une seule mesure de robustesse, pour l'ensemble des fonctions objectif originales, doit être formulée. Par exemple, Goh et Tan (2007a) proposent de calculer, pour chaque fonction objectif, le scénario dans le pire des cas, et de ne retenir que la valeur maximale sur l'ensemble des objectifs. Ainsi, seule la minimisation de la valeur correspondant au pire des cas possibles sur toutes les fonctions objectif est considérée comme mesure de robustesse. Cette dernière est alors ajoutée à l'ensemble des fonctions objectif originales. Quant à eux, Gaspar-Cunha et Covas (2008) formulent la mesure de robustesse d'une solution comme la valeur de variance moyenne ou encore la valeur de variance maximum observée sur différentes formes d'estimation de l'espérance du vecteur objectif.

4.1.2.2.3 Relations de dominance probabilistes. Cet ensemble d'approches correspond à différentes manières de modifier la relation de dominance, telle que la dominance Pareto, afin de prendre l'incertitude en compte par le biais de vecteurs objectif multiples pour chaque solution. Il suffit alors d'utiliser une tel critère au sein d'une métaheuristique basée sur une relation de dominance afin de résoudre le problème incertain considéré.
Les premières tentatives de résolution de problèmes d'optimisation multiobjectif en environnement incertain appartiennent à cette classe de métaheuristiques, et sont le fruit des travaux

respectifs de Teich (2001) et Hughes (2001), plus tard étendu par différents auteurs. Néanmoins, ces travaux préliminaires, proposés dans le cadre de fonctions objectif bruitées, font tous deux l'hypothèse que chaque solution suit une même loi de probabilité dans l'espace objectif, et supposent qu'une connaissance supplémentaire est disponible. Comme nous l'avons déjà mentionné, l'approche proposée par Teich (2001) considère que chaque élément du vecteur objectif correspond à une variable aléatoire bornée par un intervalle et suppose qu'elle suit une loi uniforme. Lorsque deux intervalles multiobjectif se chevauchent, la probabilité que l'un des deux domine l'autre est alors calculée, et la relation de dominance est vérifiée si elle est supérieure à une valeur seuil. De façon similaire, Hughes (2001) propose de calculer la probabilité qu'une solution domine une autre solution et inversement, ou encore la probabilité de prendre une mauvaise décision. Il fournit une relation de dominance pour le cas où la moyenne et la variance par objectif sont disponibles, ou lorsque seule la variance est connue. Cette relation de dominance a par exemple été expérimentée par Hughes (2001) à l'aide des algorithmes évolutionnaires MOGA et NSGA, tous deux basés sur une notion de dominance, et a par la suite été intégrée au sein de l'algorithme NSGA-II (Bui *et al.*, 2005). En règle générale, ces approches se basent sur le fait que la densité de probabilité de tous les résultats est identique, et connue ou supposée.

Fieldsend et Everson (2005) ont ultérieurement étendu le travail de Hughes (2001) en proposant des relations de dominance probabilistes dans le cas de bruit inconnu, de bruit indépendant pour chaque fonction objectif et de variance inconnue. De même, Singh (2003) a étendu le travail de Hughes (2001) pour le cas où seul un échantillon de vecteurs objectif est disponible, sans aucune information supplémentaire. Cette dernière relation de dominance probabiliste se base sur un degré de signification, donné par une distribution t de Student, pour chacune des fonctions objectif. Dans le cas où les deux échantillons sont statistiquement similaires, la solution ayant le plus petit écart type empirique est préférée. Cette notion de dominance pour des problèmes stochastiques a également été étudiée par Eskandari et Geiger (2009), mais les auteurs supposent toujours que les valeurs de chaque fonction objectif suivent une loi normale. Enfin, Trautmann *et al.* (2009) proposent une notion de dominance en présence d'incertitude basée sur l'agrégation, pour chaque fonction objectif, d'une estimation de l'espérance mathématique et de la distance entre l'enveloppe convexe de l'ensemble des vecteurs objectif vers cette estimation. Cette distance est assez semblable à une sorte d'écart type des points localisés sur la frontière de cette enveloppe convexe vers le vecteur objectif moyen, dans l'espace objectif (Trautmann *et al.*, 2009).

En référence au travail de Teich (2001), plusieurs relations de dominance ont également été proposées afin de comparer des intervalles de valeurs pour chaque fonction objectif. Dans un tel cas, une notion de dominance par intervalle monoobjectif est déjà d'ordre partiel. Ainsi, Limbourg (2005) ainsi que Limbourg et Salazar Aponte (2005) définissent les règles de dominance suivantes pour le cas monoobjectif (toujours dans le cas de minimisation).

- Un intervalle domine un autre intervalle de façon *certaine* si la borne supérieure de l'un est inférieure à la borne inférieure de l'autre.
- Un intervalle domine un autre intervalle de façon *incertaine* si les bornes inférieure et supérieure de l'un sont respectivement inférieures aux bornes inférieure et supérieure de l'autre.

Ainsi, la relation de dominance certaine est plus stricte que la relation de dominance incertaine. Et contrairement à Teich (2001), aucune distribution de probabilité n'est ici supposée entre les bornes d'un intervalle. Les auteurs étendent ensuite la dominance Pareto au cas d'intervalles multiobjectif : un intervalle multiobjectif domine un autre intervalle multiobjectif si chacune de ses composantes n'est pas dominée, et si au moins une composante domine l'autre.

4.1.2.2.4 Adaptation des valeurs de fitness. La dernière catégorie proposée consiste à adapter les valeurs de fitness affectées aux solutions afin de prendre en compte l'incertitude des vecteurs objectif. Ainsi, la stratégie d'affectation des valeurs de fitness utilisée considère directement l'ensemble des vecteurs objectif associés à chaque solution, ou à un ensemble de valeurs représentatives par objectif (typiquement, une mesure moyenne et une mesure de variance). Ce type d'approche repose généralement sur une stratégie d'affectation des valeurs de fitness déterministes qui se voit adaptée au cas stochastique.
Par exemple, Babbar *et al.* (2003) proposent d'adapter la profondeur de dominance d'une solution, utilisée au sein de l'algorithme NSGA-II, en fonction de la moyenne et de la variance observées sur l'ensemble des évaluations pour chacune des fonctions objectif. Ainsi, une solution qui n'appartient pas au premier front au vu des vecteurs objectif moyens est tout de même intégrée à cet ensemble si elle se trouve proche d'une solution de rang 1 dans l'espace objectif. Quant à Barrico et Antunes (2007), ils utilisent également la profondeur de dominance pour calculer les valeurs de fitness des solutions. Cette étape initiale se base sur les vecteurs objectif réels, car les auteurs traitent d'incertitude localisée sur les variables de décision. Mais la valeur de fitness d'une solution est ensuite incrémentée d'un *degré de robustesse*, calculée en fonction de la qualité des solutions perturbées par rapport à l'originale en termes de dominance Pareto, de sorte qu'un ordre total semble dès lors vraisemblablement défini entre les solutions.

4.1.2.3 Questions connexes

Une des questions fondamentales liée à la résolution d'un problème d'optimisation multiobjectif stochastique par la considération d'un ensemble de vecteurs n'a pas encore été tranchée. Elle concerne le choix d'une taille appropriée de l'échantillon de vecteurs objectif associé à chaque solution, et si cette taille doit être fixe ou adaptée au cours de la recherche, ou encore selon la solution considérée. Aussi, plutôt que de réévaluer une même solution à plusieurs reprises, une autre technique semblable consiste à n'affecter qu'un seul vecteur objectif par solution, tout en manipulant une population de très grande taille au sein de la métaheuristique développée. En effet, lorsque la taille de la population est extrêmement large, beaucoup de solutions identiques sont généralement observées, chacune d'elle ayant vraisemblablement un vecteur objectif différent de par la nature stochastique de la fonction d'évaluation. De cette façon, l'influence de l'incertitude est atténuée, et cette stratégie résulte en un sens à un calcul de moyenne implicite (Tan et Goh, 2008). De même, une autre approche consiste à ne conserver qu'un seul vecteur objectif par solution, mais à réévaluer cette même solution régulièrement, ceci afin de limiter l'impact de solutions dont les valeurs objectif sont aberrantes (Büche *et al.*, 2002). Par ailleurs, Bui *et al.* (2005) proposent qu'une nouvelle solution hérite de la valeur des vecteurs objectif de ses parents, ceci dans le cadre d'un algorithme évolutionnaire. Chaque solution est évaluée un petit nombre de fois, et si les vecteurs objectif obtenus diffèrent trop de ceux des parents, elle est réévaluée à de plus grandes reprises, sinon elle hérite des vecteurs objectif de ses parents. Ceci permet de sauver un temps de calcul précieux lorsque la fonction d'évaluation est trop coûteuse à calculer. D'autre part, remarquons que Goh *et al.* (2007) ont proposé divers benchmarks de problèmes stochastiques continus, et ont étudié l'impact de fonctions objectif bruitées sur les performances d'un certain nombre d'algorithmes évolutionnaires multiobjectif (Goh et Tan, 2007b).

Pour résumer, un nombre restreint mais croissant d'études sont consacrées à la résolution de problèmes d'optimisation multiobjectif en environnement incertain à l'aide de métaheuristiques, pour

l'essentiel des algorithmes évolutionnaires. Une première remarque est que plusieurs approches existantes supposent généralement des caractéristiques spécifiques concernant la distribution de probabilité associée à une fonction objectif donnée. Par conséquent, ce type de méthode exploite des connaissances qui peuvent ne pas être disponibles en pratique. Une seconde remarque est que les méthodes proposées dans la littérature ont presque toutes été expérimentées sur des problèmes d'optimisation continue. Par ailleurs, selon la classification proposée par Jin et Branke (2005), elles ont majoritairement été appliquées à des problèmes pour lesquels les fonctions objectif sont bruitées, ou lorsque des solutions robustes vis-à-vis des variables de décision sont recherchées. Une dernière remarque concerne la question clé de l'évaluation des performances des méthodes de recherche en environnement incertain, aucun protocole approprié ayant été proposé jusqu'à présent. Nous reviendrons sur ce dernier point lors de l'analyse expérimentale.
Par la suite, nous présentons un ensemble de méthodes basées sur un indicateur de qualité afin de résoudre des problèmes d'optimisation multiobjectif stochastiques, où une distribution de probabilité arbitraire est associée à chaque solution dans l'espace objectif. Nous essayerons aussi de commenter un certain nombre de questions ouvertes et fondamentales, relatives à l'évaluation des performances des méthodes de recherche multiobjectif en environnement incertain. Pour finir, même si les contributions de cette thèse peuvent potentiellement être appliquées à un grand nombre de problèmes d'optimisation multiobjectif stochastique, nous les expérimenterons sur un problème combinatoire pour lequel les paramètres environnementaux sont sujets à incertitude.

4.2 Conception

Suite à une modélisation plus formelle de la prise en compte de l'incertitude, nous proposons un ensemble de nouvelles approches dédiées à la résolution de problèmes d'optimisation multiobjectif stochastiques.

4.2.1 Modélisation de l'incertitude

Dans le cadre de problèmes d'optimisation multiobjectif déterministes, un vecteur objectif unique $z \in Z$ est strictement affecté à un vecteur de décision $x \in X$ sur la base de la fonction objectif f. Ainsi, $f(x)$ représente l'évaluation réelle de x, et f représente donc une application déterministe de l'ensemble des solutions réalisable de l'espace décisionnel X vers l'ensemble des vecteurs réalisables dans l'espace objectif Z. Lors de la prise en compte de l'incertitude, à chaque solution est associé un ensemble de vecteurs objectif, chacun d'eux résultant éventuellement en un point différent de l'espace objectif. Nous allons donc considérer que f ne représente pas une application déterministe de X vers Z, mais plutôt qu'un ensemble potentiellement infini de vecteurs objectif est dorénavant associé à une solution réalisable x. Nous supposons que l'évaluation « réelle » d'une solution est absolument inconnue avant la fin du processus de recherche. Aucune hypothèse n'est formulée sur une quelconque distribution de probabilité associée aux fonctions objectif, aux variables de décisions, ou aux paramètres environnementaux ; une telle distribution étant généralement inconnue à l'avance et étant susceptible de différer pour toute solution.
Ainsi, dans le cas incertain, à chaque solution $x \in X$ est associé un échantillon de vecteurs objectif. Plus élevé est le degré d'incertitude, plus grande sera la variance observée sur les vecteurs objectif résultant de l'évaluation multiple de x. Il s'avère donc nécessaire de déterminer une taille d'échantillon satisfaisante, car l'étape d'échantillonnage peut s'avérer coûteuse en temps de calcul.

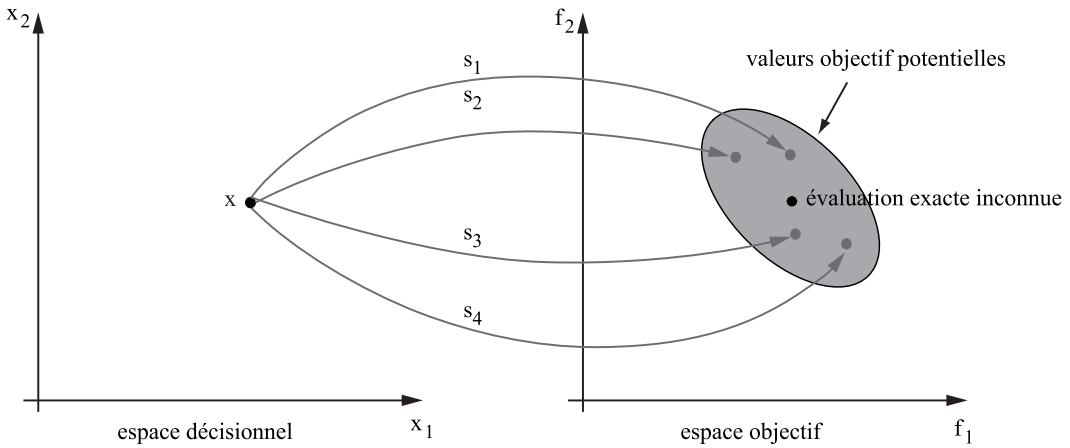


Fig. 4.6 – Illustration de l'évaluation multiple d'une solution x à partir de quatre scénarios différents $\{s_1, s_2, s_3, s_4\}$.

En effet, l'évaluation d'une solution étant parfois très longue, il faut trouver un bon compromis entre une précision fine et un temps de calcul raisonnable. De surcroît, le nombre d'évaluations possibles pour une solution est généralement limité en pratique. Dans ce cas, il est important de n'utiliser qu'une sous-partie de ces évaluations au cours du processus de recherche, la partie restante se montrant généralement nécessaire lors de l'évaluation des performances du, ou des algorithmes considérés.
Plus formellement, à toute solution $x \in X$, nous supposons qu'un ensemble d'évaluations indépendantes et équiprobables est calculé. Ainsi, un échantillon de vecteurs objectif $\{z^{(i)}\}_{i=1}^{p}$ est désormais associé à chaque solution. Maintenant, deux cas peuvent se présenter. Premièrement, pour une évaluation donnée d'une solution arbitrairement considérée, il se peut que la fonction d'évaluation soit strictement indépendante de toutes les évaluations réalisées jusqu'à présent. Dans ce cas, tous les vecteurs objectif de toutes les solutions évaluées sont strictement indépendants les uns des autres, et la taille de l'échantillon de vecteurs objectif peut éventuellement différer d'une solution à l'autre. Une autre alternative justifiable est de considérer un ensemble fini de scénarios indépendants et équiprobables $S = \{s_1, s_2, \ldots, s_p\}$ (Kouvelis et Yu, 1997). $S(x) = \{z^{(1)}, z^{(2)}, \ldots, z^{(p)}\}$ correspond à l'échantillon d'évaluations indépendantes relatives à une solution $x \in X$. Dans ce cas, l'élément $z^{(i)}$ de l'échantillon associé à x représente la valeur du vecteur objectif de x si le scénario s_i se produit (Fig. 4.6). Par cela, nous supposons que, étant donné deux solutions x et $x' \in X$, les échantillons de vecteurs objectif $\{z^{(i)}\}_{i=1}^{p}$ et $\{z'^{(j)}\}_{j=1}^{p'}$ sont appariés, et de même taille ($p = p'$). Ainsi, pour un scénario s_i donné, les vecteurs objectif correspondants $z^{(i)}$ et $z'^{(i)}$ sont fondamentalement comparables l'un à l'autre.

4.2.2 Approches métaheuristiques pour l'optimisation multiobjectif en environnement incertain

Les algorithmes que nous proposons pour la résolution de problèmes d'optimisation multiobjectif stochastiques se basent tous sur la stratégie d'affectation de valeurs de fitness basée sur un indicateur binaire de qualité proposée par Zitzler et Künzli (2004), et introduite à la section 2.1.3.1. Afin de prendre en compte l'incertitude, nous introduisons un ensemble de huit indicateurs de

qualité permettant de manipuler l'incertitude, et pouvant être utilisés au sein de n'importe quelle métaheuristique basée sur un indicateur binaire, comme IBEA (Zitzler et Künzli, 2004) ou encore IBMOLS (Basseur et Burke, 2007). Ces indicateurs correspondent en quelque sorte à différentes stratégies parmi lesquelles le décideur peut choisir selon ses préférences, ou selon le type de problème à résoudre. Nous supposons tout d'abord qu'un indicateur binaire $I : Z \times Z \rightarrow \mathbb{R}$, dédié à la comparaison de deux vecteurs objectif, est défini. Deux exemples ont déjà été donnés à la section 2.1.4.3, à savoir les indicateurs $I_{\epsilon+}$ et I_{HD}. Aussi, contrairement au cas déterministe, l'interprétation de ce type d'indicateur depuis l'espace objectif vers l'espace décisionnel n'est plus immédiate. En effet, à une solution de l'espace décisionnel ne correspond plus un vecteur objectif unique, mais plutôt un ensemble de vecteurs objectif.

$$I(x, x') \neq I(f(x), f(x')) \tag{4.1}$$

Les approches proposées ci-dessous consistent donc à définir différentes stratégies d'agrégation de l'information donnée par deux échantillons de vecteurs objectif $\{z_k^{(1)}, z_k^{(2)}, \ldots, z_k^{(p)}\}$ et $\{z_k'^{(1)}, z_k'^{(2)}, \ldots, z_k'^{(p')}\}$, respectivement associés à deux solutions x et $x' \in X$, en une I-valeur scalaire unique. En ce sens, ce travail étend donc la contribution de Basseur et Zitzler (2006) qui ont déjà proposé de tels indicateurs pour traiter de fonctions objectif bruitées, et pour le cas particulier de l'indicateur $I_{\epsilon+}$. Les approches présentées ici sont indépendantes de l'indicateur $I : Z \times Z \rightarrow \mathbb{R}$ choisi. La I-valeur obtenue peut donc tout naturellement prendre place au sein de la stratégie d'affectation d'une valeur de fitness basée sur un indicateur binaire de qualité. Pour mémoire, le calcul des valeurs de fitness est rappelé ci-dessous.

$$F(x) = \sum_{x' \in P \setminus \{x\}} -e^{-I(x',x)/\kappa} \tag{4.2}$$

Deux types d'indicateur, correspondant à deux niveaux de scalarisation, sont ici proposés (Fig.4.7). Tout d'abord, les indicateurs situés au niveau des valeurs objectifs se basent directement sur l'échantillon $\{z_k^{(1)}, z_k^{(2)}, \ldots, z_k^{(p)}\}$ associé à une solution $x \in X$. Ces approches consistent à transformer l'échantillon associé à une solution en un vecteur objectif unique par le biais d'une valeur, jugée représentative, par fonction objectif. Ces approches situées au niveau des vecteurs objectif sont basées sur des concepts génériques, et peuvent donc être potentiellement appliquées à tout type de méthode pour l'optimisation multiobjectif, et pas uniquement à celles qui sont basées sur un indicateur binaire de qualité. Pour cela, il suffit de considérer le vecteur représentatif unique choisi comme vecteur objectif déterministe. Le deuxième type d'indicateur, situés au niveau des I-valeurs, consistent à calculer, dans un premier temps, un échantillon de I-valeurs issu de deux échantillons de vecteurs objectif et associé à une paire de solutions de la population courante. Au sein de ces deux classes d'approches, quatre indicateurs sont proposés. Ils correspondent respectivement à une stratégie dans le meilleur des cas, dans le pire des cas, dans le cas moyen et dans le cas médian (Fig. 4.8).

4.2.2.1 Approches basées sur les vecteurs objectif

Le premier ensemble d'indicateurs défini dans le but de manipuler l'incertitude porte sur le niveau des vecteurs objectif. Ils sont directement basés sur l'échantillon $\{z_k^{(1)}, z_k^{(2)}, \ldots, z_k^{(p)}\}$ associé à une solution $x \in X$. Les concepts introduits ici sont généraux, car ils peuvent être utilisés au

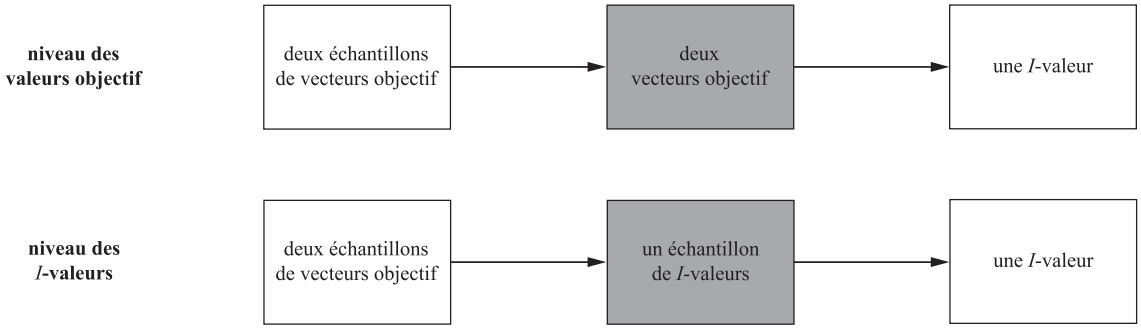


Fig. 4.7 – Deux classes de métaheuristiques proposées pour la résolution de problèmes multiobjectif stochastiques : les approches basées au niveau des valeurs objectif, et les approches basées au niveau des I-valeurs.

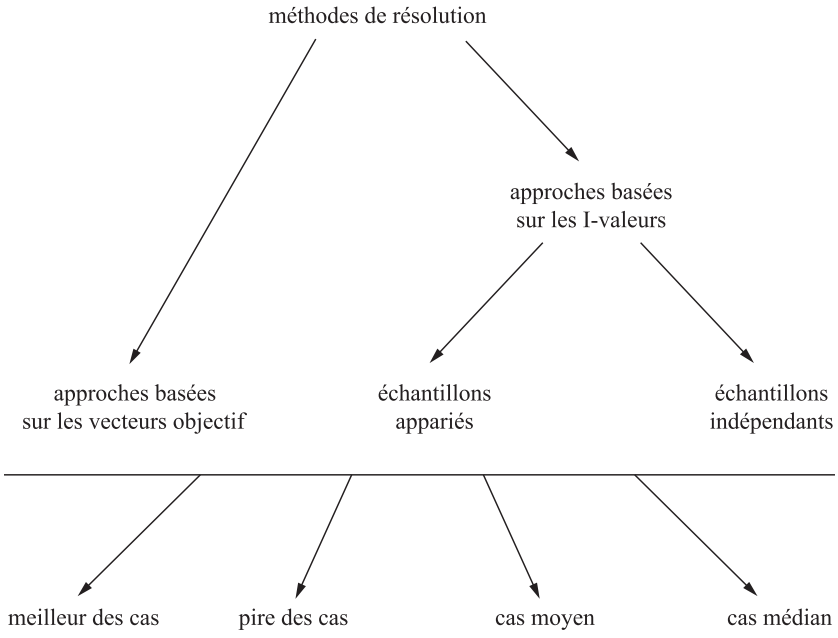


Fig. 4.8 – Résumé des méthodes de résolution proposées.

sein de n'importe quelle métaheuristique pour l'optimisation multiobjectif. Leur utilisation au sein d'une stratégie d'affectation des valeurs de fitness basée sur un indicateur binaire consiste donc à transformer l'échantillon de vecteurs objectif d'une solution en un vecteur objectif unique. Ensuite, les vecteurs objectif transformés associés à deux solutions sont comparés à l'aide d'un indicateur binaire localisé au sein de l'espace objectif : $I : Z \times Z \rightarrow \mathbb{R}$.

4.2.2.1.1 Meilleur des cas. Ce premier indicateur ($I^{z^{best}}$) est fondé sur un vecteur objectif dans le meilleur des cas (z^{best}) calculé à partir de l'échantillon de vecteurs objectif associé à une solution. Pour chaque fonction objectif, la valeur observée la plus efficace est retenue, et le vecteur objectif correspondant est défini (4.3), où n est le nombre de fonctions objectif [1]. z^{best} est alors considéré comme le vecteur objectif unique d'un problème déterministe, et une approche de résolution classique peut être employée en utilisant une telle évaluation.

$$I^{z^{best}}(x, x') = \begin{cases} I(z^{best}, z'^{best}) \\ \text{tel que} \quad z_k^{best} = \min_{i \in \{1,\ldots,p\}} z_k^{(i)} \quad \forall k \in \{1, \ldots, n\} \end{cases} \tag{4.3}$$

Cela conduit à une stratégie optimiste, ce qui signifie que le processus de recherche ignore le scénario effectif correspondant au pire des cas.

4.2.2.1.2 Pire des cas. À l'inverse, un vecteur objectif dans le pire des cas (z^{worst}) peut être défini comme suit, ce qui résulte en une approche pessimiste, conservatrice, et ayant une haute aversion au risque. Celle-ci est donc liée au domaine de l'optimisation robuste (Kouvelis et Yu, 1997).

$$I^{z^{worst}}(x, x') = \begin{cases} I(z^{worst}, z'^{worst}) \\ \text{tel que} \quad z_k^{worst} = \max_{i \in \{1,\ldots,p\}} z_k^{(i)} \quad \forall k \in \{1, \ldots, n\} \end{cases} \tag{4.4}$$

4.2.2.1.3 Cas moyen. Lorsqu'elles prennent en compte l'incertitude, la plupart des approches visent à atteindre une valeur optimale de l'espérance mathématique (ou un ensemble de valeurs dans le cadre de l'optimisation multiobjectif). Ainsi, la performance moyenne d'une solution est une indication importante de sa qualité. Laissez-nous définir un vecteur objectif dans le cas moyen (z^{avg}) ainsi qu'un indicateur de qualité correspondant ($I^{z^{avg}}$).

$$I^{z^{avg}}(x, x') = \begin{cases} I(z^{avg}, z'^{avg}) \\ \text{tel que} \quad z_k^{avg} = \frac{1}{p} \sum_{i=1}^{p} z_k^{(i)} \quad \forall k \in \{1, \ldots, n\} \end{cases} \tag{4.5}$$

Bien sûr, des moyennes différentes de la moyenne arithmétique peuvent également être considérées. Un tel cas moyen est plus que couramment utilisée en optimisation monoobjectif (Jin et Branke, 2005), et a déjà été considérée dans plusieurs études liées à l'optimisation multiobjectif (Babbar *et al.*, 2003; Deb et Gupta, 2006).

1. Rappelons qu'un problème de minimisation est ici supposé.

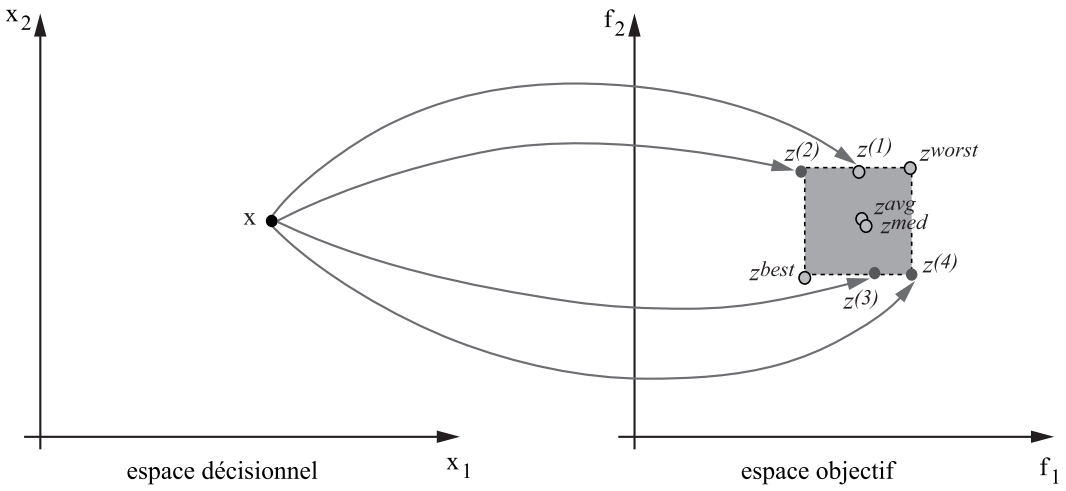


Fig. 4.9 – Illustration des différents vecteurs objectif représentatifs considérés par les approches basées au niveau des valeurs objectif : le meilleur des cas (z^{best}), le pire des cas (z^{worst}), le cas moyen (z^{avg}) et le cas médian (z^{med}).

4.2.2.1.4 Cas médian. Dans certains cas, il se peut qu'une valeur médiane se montre plus pertinente que la valeur moyenne pour le problème traité. C'est la raison pour laquelle nous introduisons également un vecteur objectif dans le cas médian (z^{med}).

$$I^{z^{med}}(x,x') = \begin{cases} I(z^{med}, z'^{med}) \\ \text{tel que } \; z_k^{med} \; \text{ est la médiane de } \; \{z_k^{(1)}, z_k^{(2)}, \ldots, z_k^{(p)}\} \;\; \forall k \in \{1,\ldots,n\} \end{cases} \tag{4.6}$$

Les vecteurs objectif transformés correspondant aux quatre cas présentés ci-dessus sont illustrés sur la figure 4.9.

4.2.2.2 Approches basées sur les valeurs d'indicateur

Contrairement aux stratégies précédentes, les quatre indicateurs présentés ci-dessous se basent sur les I-valeurs obtenues à partir des échantillons de vecteurs objectif de deux solutions. Ils sont donc spécifiques aux méthodes basées sur un indicateur binaire de qualité.
Considérons donc deux solutions arbitraires x et $x' \in X$ issues de la population courante, ainsi que leurs échantillons de vecteurs objectif $\{z_k^{(1)}, z_k^{(2)}, \ldots, z_k^{(p)}\}$ et $\{z_k'^{(1)}, z_k'^{(2)}, \ldots, z_k'^{(p')}\}$ correspondants. Pour chacun des indicateurs proposés ci-dessous, deux cas peuvent se présenter. Soit le scénario considéré lors de la $i$ème évaluation des deux solutions est identique, et ceci pour chaque $i \in \{1,2,\ldots,p\}$ (avec $p = p'$), auquel cas les échantillons de vecteurs objectif associés à ces deux solutions sont *appariés*. Soit, au contraire, les scénarios sont différents ou aléatoirement choisis à chaque évaluation, auquel cas les échantillons de vecteurs objectif sont *indépendants*.
Comme proposé par Basseur et Zitzler (2006), les approches proposées consistent à définir une stratégie d'agrégation à partir de l'échantillon de I-valeurs obtenu de la comparaison deux à deux des vecteurs objectif des deux solutions x et x'. Dans le cas d'échantillons appariés, la stratégie considérée se basera sur l'échantillon de I-valeurs suivant, de taille p, et où $I : Z \times Z \rightarrow \mathbb{R}$ est

un indicateur arbitraire.

$$I(z^{(1)}, z'^{(1)}), I(z^{(2)}, z'^{(2)}), \ldots, I(z^{(p)}, z'^{(p)}) \tag{4.7}$$

Dans le cas d'échantillons indépendants, l'échantillon considéré sera plutôt l'ensemble suivant, de taille $(p \times p')$.

$$I(z^{(1)}, z'^{(1)}), I(z^{(1)}, z'^{(2)}), \ldots, I(z^{(i)}, z'^{(j)}), \ldots, I(z^{(p)}, z'^{(p')}) \tag{4.8}$$

Ainsi, l'idée de ce deuxième type d'approche est de considérer toutes les combinaisons de vecteurs objectif, au contraire des méthodes précédentes dont les caractéristiques de distribution sont probablement perdues.
De façon semblable aux indicateurs précédents, les quatre stratégies proposées qui se basent sur les I-valeurs correspondent au meilleur des cas, au pire des cas, au cas moyen et au cas médian provenant de la comparaison par paire, selon un indicateur I considéré, des vecteurs objectif associés à deux solutions. Sans perte de généralité, nous considérerons par la suite que les I-valeurs sont à minimiser.

4.2.2.2.1 Meilleur des cas. Étant donné un indicateur I et deux solutions x et x' issues de la population courante, une approche possible visant à exploiter l'information contenue au sein des I-valeurs consiste à considérer le meilleur élément de l'échantillon de I-valeurs. La stratégie résultante est une stratégie optimiste où seul le vecteur objectif associé à x qui semble le plus prometteur vis-à-vis de x' est pris en compte.

$$I^{best}(x, x') = \begin{cases} \min_{i \in \{1,\ldots,p\}} & I(z^{(i)}, z'^{(i)}) & \text{si les échantillons sont appariés} \\ \min_{i \in \{1,\ldots,p\}, j \in \{1,\ldots,p'\}} & I(z^{(i)}, z'^{(j)}) & \text{si les échantillons sont indépendants} \end{cases} \tag{4.9}$$

4.2.2.2.2 Pire des cas. Réciproquement, un indicateur pessimiste peut être défini comme suit.

$$I^{worst}(x, x') = \begin{cases} \max_{i \in \{1,\ldots,p\}} & I(z^{(i)}, z'^{(i)}) & \text{si les échantillons sont appariés} \\ \max_{i \in \{1,\ldots,p\}, j \in \{1,\ldots,p'\}} & I(z^{(i)}, z'^{(j)}) & \text{si les échantillons sont indépendants} \end{cases} \tag{4.10}$$

4.2.2.2.3 Cas moyen. De même, permettez-nous de spécifier un indicateur reflétant le cas moyen, où la moyenne des I-valeurs est employée dans le calcul de fitness de la solution.

$$I^{avg}(x, x') = \begin{cases} \frac{1}{p} \sum_{i=1}^{p} & I(z^{(i)}, z'^{(i)}) & \text{si les échantillons sont appariés} \\ \frac{1}{p \cdot p'} \sum_{i=1}^{p} \sum_{j=1}^{p'} & I(z^{(i)}, z'^{(j)}) & \text{si les échantillons sont indépendants} \end{cases} \tag{4.11}$$

Dans le cas d'échantillons de vecteurs objectif appariés, cette stratégie est identique à une stratégie additive à un facteur multiplicatif près, correspondant au nombre d'évaluations par solution.

4.2.2.2.4 Cas médian. Enfin, un indicateur représentant le cas médian est une dernière possibilité. Pour cette stratégie est considérée la valeur médiane de l'ensemble (4.7) en cas d'échantillons appariés, et de l'ensemble (4.8) en cas d'échantillons indépendants.

Les huit indicateurs proposés pour la prise en compte de l'incertitude peuvent désormais prendre place au sein de la stratégie d'affectation des valeurs de fitness basée sur un indicateur binaire, et donc au sein de métaheuristiques comme IBEA. Ils donnent donc naissance à un ensemble d'algorithme évolutionnaires capables de résoudre un problème d'optimisation multiobjectif en environnement incertain. Ces stratégies permettent de préciser différents types de préférence en termes de prise en compte de l'incertitude, ceci par simple définition d'un indicateur. Ainsi, seuls deux niveaux sont dorénavant à déterminer lors du choix d'instanciation de l'algorithme IBEA pour résoudre un problème stochastique : un indicateur pour la comparaison de deux vecteurs objectif (tout comme dans le cas déterministe), et un indicateur additionnel pour la prise en compte de l'incertitude.

4.3 Implémentation

Les algorithmes présentés dans la section précédente, ainsi que les méthodologies de modélisation de l'incertitude, seront bientôt disponibles au sein de la plateforme ParadisEO-MOEO.
Par rapport à une métaheuristique classique pour l'optimisation multiobjectif déterministe, veuillez noter que l'effort d'implémentation est relativement restreint. En effet, la fonction d'évaluation doit être adaptée au cas où plus d'un seul vecteur objectif est calculé par solution. Aussi, par rapport à l'affectation des valeurs de fitness, seul un indicateur de qualité supplémentaire, basé sur la comparaison d'un ensemble de vecteurs objectif, doit être défini pour pouvoir être utilisé au sein d'une stratégie d'affectation des valeurs de fitness, et donc au sein de métaheuristiques comme IBEA ou IBMOLS.

4.4 Analyse expérimentale

Cette section traite de l'évaluation des performances de méthodes pour la résolution de problèmes d'optimisation multiobjectif sous incertitude. Dans un second temps, un problème de Flowshop stochastique est formulé, et les différentes métaheuristiques proposées y sont appliquées et expérimentées.

4.4.1 Évaluation des performances

Comme nous l'avons déjà remarqué dans le premier chapitre, dans le cas déterministe, trouver une approximation de l'ensemble Pareto optimal est en soi un problème biobjectif, puisque le but de l'optimisation consiste à obtenir un ensemble de solutions présentant à la fois de bonnes propriétés en termes de convergence et de diversité. Cette question a été relativement étudiée dans la littérature (Zitzler *et al.*, 2003, 2008). Pourtant, lors de la prise en compte de l'incertitude, l'évaluation des performances n'a pas été abordée de façon satisfaisante à ce jour.

4.4.1.1 Commentaires généraux

Une pratique courante se contente de convertir le problème stochastique à résoudre en un problème déterministe par le biais de telle ou telle stratégie, et d'évaluer les performances des résultats par rapport à cette formulation déterministe. Par exemple, la moyenne observée sur un échantillon de vecteurs objectif peut être utilisée (Teich, 2001; Tan *et al.*, 2007; Eskandari et Geiger, 2009). Un autre exemple souvent rencontré consiste à négliger purement et simplement l'incertitude encore prise en compte lors de l'étape de recherche, et à considérer le modèle déterministe duquel le problème stochastique étudié a été tiré comme la « véritable » réalisation des données incertaines (Fieldsend et Everson, 2005; Goh et Tan, 2007b; Goh *et al.*, 2007). Comme souligné précédemment, dans les situations du monde réel, il n'existe pas d'évaluation unique (ni un cas moyen, ni un cas « réel ») associée à une solution donnée, ou tout au moins, celle-ci n'est généralement pas connue à l'avance. Ainsi, aucune réalisation des données stochastiques ne peut être considérée plus plausible qu'une autre, qu'elle soit issue de données déterministes, ou d'une réalisation arbitrairement sélectionnée. Tout comme lors de l'étape de recherche, il nous semble important de respecter les points suivants :

- À la suite du processus de recherche, il est préférable de considérer un échantillon d'évaluations possibles plutôt qu'une évaluation unique afin d'évaluer les performances d'une solution, ou d'un ensemble de solutions obtenues par un algorithme donné.
- Il nous paraît par ailleurs pertinent d'envisager de réévaluer les solutions obtenues à l'aide de données incertaines qui diffèrent de celles qui ont été utilisées durant le processus de recherche, ceci afin de supprimer l'influence du processus de recherche, et donc de fournir une comparaison équitable et non biaisée entre les algorithmes étudiés.

4.4.1.2 Deux approches méthodologiques

Les deux approches méthodologiques que nous proposons se basent sur des idées similaires à celles que nous avons présentées pour les méthodes de résolution dans la section précédente. Ainsi, la première classe d'approches se base sur les vecteurs objectif alors que la deuxième classe se base sur les I-valeurs.

4.4.1.2.1 Approches basées sur une évaluation représentative. Soit A un ensemble des solutions trouvées par un algorithme donné. À chaque solution $x \in A$ est donc associé un ensemble de vecteurs objectif $\{z_k^{(1)}, z_k^{(2)}, \ldots, z_k^{(q)}\}$ de taille q, q étant donc le nombre de réalisations stochastiques considérés. La première approche consiste à calculer, pour chaque solution, un vecteur objectif unique correspondant au meilleur des cas (z^{best}), au pire des cas (z^{worst}), au cas moyen (z^{avg}) et au cas médian (z^{med}), comme expliqué à la section précédente. Ainsi à chaque solution est dorénavant associé un vecteur objectif unique reflétant quatre cas préférentiels vis-à-vis de la prise en compte de l'incertitude. Un protocole d'évaluation classique peut donc être utilisé. Il va déterminer si, pour une instance donnée, un algorithme A et un algorithme B, exécutés chacun un même nombre de fois, ont obtenu des résultats significativement différents, et le cas échéant si l'un des deux a obtenu des résultats statistiquement plus performants que l'autre, toujours selon un indicateur de qualité I spécifié (hypervolume et epsilon), et désormais selon les quatre cas préférentiels susmentionnés.
Il semble donc raisonnable de penser que la première classe de méthodes de résolutions proposée, qui utilise ce même type d'évaluation représentative, se comportera de façon performante quant

à ce protocole d'évaluation.

4.4.1.2.2 Approches basées sur les valeurs d'indicateur. La seconde méthodologie d'évaluation des performances se base sur les valeurs d'indicateur obtenues par les différentes approximations trouvées par les algorithmes (et non sur celles obtenues par les solutions comme c'était le cas à la section précédente). Soient A et B deux ensembles des solutions trouvées par deux méthodes de résolution différentes pour un même problème. À chaque solution $x \in A$ est donc associé un ensemble de vecteurs objectif $\{z_k^{(1)}, z_k^{(2)}, \ldots, z_k^{(q)}\}$ de taille q, et à chaque solution $x' \in B$ est donc associé un ensemble de vecteurs objectif $\{z_k'^{(1)}, z_k'^{(2)}, \ldots, z_k'^{(q')}\}$ de taille q'. Ainsi, une fois encore, deux cas peuvent se présenter : soit les échantillons de vecteurs objectif sont appariés (auquel cas $q = q'$), soit ils sont indépendants. S'ils sont appariés, cela signifie que q scénarios $S = \{s_1, s_2, \ldots, s_q\}$ ont été considérés, et donc que les algorithmes considérés peuvent être comparés scénario par scénario. C'est ce deuxième cas que nous allons considérer par la suite. Néanmoins, notez que le protocole proposé peut facilement être adapté au cas où les échantillons seraient indépendants.
Limitons nous tout d'abord au cas où un seul scénario $s \in S$ est considéré, et où un algorithme A a obtenu r approximations, issues de r exécutions indépendantes de l'algorithme. Relativement à un indicateur I donné, une analyse déterministe peut être effectuée. Nous obtenons donc un ensemble de r valeurs scalaires $\{I(A_1), I(A_2), \ldots, I(A_r)\}$, où $I(A_i)$ correspond à la valeur obtenue par la $i$ème exécution de l'algorithme A selon l'indicateur I, et toujours pour le scénario s. Maintenant, si nous considérions les q scénarios simultanément, nous obtenons donc l'ensemble de I-valeurs suivants :

	exécution 1	exécution 2	$\cdots$	exécution r
scénario s_1	$I(A_1^1)$	$I(A_2^1)$	$\cdots$	$I(A_r^1)$
scénario s_2	$I(A_1^2)$	$I(A_2^2)$	$\cdots$	$I(A_r^2)$
$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$
scénario s_q	$I(A_1^q)$	$I(A_2^q)$	$\cdots$	$I(A_r^q)$

Pour une exécution j donnée, il nous reste à calculer la I-valeur dans le meilleur des cas, le pire des cas, le cas moyen et le cas médian de l'ensemble suivant : $\{I(A_j^1), I(A_j^2), \ldots, I(A_j^r)\}$. Et ceci pour chacun des algorithmes considérés. Ainsi, une seule I-valeur scalaire est finalement obtenue par algorithme et par scénario. Nous pouvons donc utiliser le même test statistique que pour le cas déterministe afin de dévoiler si un algorithme est significativement meilleur qu'un autre ou non, ceci selon l'ensemble des scénarios considérés, selon les quatre cas préférentiels considérés (meilleur des cas, pire des cas, cas moyen et cas médian), et toujours selon un indicateur I donné. Lors de nos expérimentations, nous allons toujours considérer les indicateurs epsilon et hypervolume, et un test statistique non-paramétrique de Wilcoxon, les échantillons considérés étant supposés appariés.

4.4.2 Application au problème de Flowshop avec durées d'exécutions stochastiques

Dans cette section, nous présentons un modèle stochastique du problème de Flowshop biobjectif, pour lequel il s'agit de trouver des solutions robustes vis-à-vis de paramètres environnementaux

incertains. Nous nous intéressons aux différentes sources d'incertitude auxquelles il peut être soumis. Nous nous focalisons ensuite sur le problème de Flowshop dont les durées d'exécutions sont incertaines et sont modélisées à l'aide de variables aléatoires. L'incertitude est donc prise en compte à l'aide de différentes versions, réalisations, ou scénarios, des valeurs de paramètres. Divers jeux de données sont proposés et mis à disposition sur le web. Enfin, nous adaptons les différentes méthodologies algorithmiques à la résolution du problème, et nous appliquons différents protocoles d'évaluation des performances pour mesurer le comportement des approches proposées. Bien que ce travail se concentre sur le problème de Flowshop, il est parfaitement généralisable à tout type de problème d'optimisation multiobjectif stochastique et aisément adaptable à d'autres problèmes d'ordonnancement.

4.4.2.1 Sources d'incertitude

Dans les situations réelles d'ordonnancement, l'incertitude provient principalement des variables d'environnement et peut donc être classée dans la seconde catégorie de la classification proposée par Jin et Branke (2005) : variables de décisions ou paramètres environnementaux soumis à des perturbations ; voir la section 4.1.1.1. C'est en cela que réside l'originalité de l'approche proposée, car, à notre connaissance, aucune étude n'a été menée sur un problème d'optimisation multiobjectif dont les paramètres environnementaux sont incertains par le biais de métaheuristiques. Or, rares sont les paramètres associés aux problèmes d'ordonnancement qui sont dépourvus d'incertitude. Les solutions sont sensibles à ces perturbations, et il s'avère donc indispensable de tenir compte des « zones d'ignorances » et des « à-peu-près » que laissent entrevoir certains paramètres (Roy, 2005). Cette incertitude peut provenir de différentes sources, telles que des variations sur les dates dues, des pannes de machines, l'ajout ou la suppression inattendue de jobs, des durées d'exécutions variables, etc. D'après la littérature du domaine, il apparaît que pour le Flowshop de permutation étudié ici, l'incertitude est susceptible de provenir essentiellement des dates dues et des durées d'exécutions.
Tout d'abord, dans le modèle déterministe, la date due d'un job J_i est donnée par une constante d_i. Pourtant, déterminer ce nombre sans ambiguïté paraît difficile. Il semblerait donc plus naturel de donner un intervalle $[d_i^1, d_i^2]$ durant lequel la satisfaction humaine pour la complétion du job J_i décroît. De plus, une date de fin d_i peut subir des modifications dynamiques du fait qu'un job sans trop d'importance à un moment peut devenir plus important à un autre moment (et vice versa). Par ailleurs, la durée d'exécution d'une tâche sur une machine peut varier d'une exécution à une autre du fait d'événements extérieurs. Ainsi, la durée d'exécution p_{ij} associée à une tâche t_{ij} est rarement constante en pratique. Il paraît donc évident qu'aucun paramètre ne peut être vu comme une donnée exacte et précise et que des modèles non déterministes doivent être élaborés pour résoudre un problème d'ordonnancement. Pour cela, nous décidons d'adopter une approche pro-active où les durées d'exécutions sont considérées comme incertaines et sont modélisées à l'aide de variables aléatoires.

4.4.2.2 Modèles stochastiques et jeux de données

À notre connaissance, le problème du Flowshop stochastique n'a jamais été étudié de façon multiobjectif. Pourtant, dans un cas réel, dès qu'un historique des temps d'exécution précis et valide des différentes tâches est disponible, il est relativement aisé d'obtenir la distribution de probabilité associée aux données et, ainsi, la loi de probabilité correspondante. Suite à une analyse, nous

proposons ici quatre distributions de probabilité générales susceptibles d'être suivies par une durée d'exécution aléatoire. Bien sûr, une analyse statistique rigoureuse, basée sur des données réelles, est impérative pour déterminer la loi de distribution exacte associée à une durée d'exécution pour un problème concret d'ordonnancement. Les lois de probabilités énoncées ci-dessous ont été décidées afin de coller au mieux à la réalité, et donc de fournir une bonne illustration de ce qui peut typiquement être trouvé dans le monde industriel. Une revue du problème du Flowshop monoobjectif avec durées d'exécutions aléatoires est proposé par Gourgand *et al.* (2005).

4.4.2.2.1 Loi uniforme. Tout d'abord, il se peut qu'une durée d'exécution p_{ij} soit comprise entre deux valeurs a et b, la répartition étant uniforme entre ces deux valeurs (voir figure 4.10). p_{ij} suit alors une *loi uniforme* sur l'intervalle $[a, b]$ et admet pour densité :

$$pdf(x) = \begin{cases} \frac{1}{b-a} & \text{si } x \in [a, b] \\ 0 & \text{sinon} \end{cases} \tag{4.12}$$

Bien que peu d'exemples de durées d'exécutions suivant une loi uniforme aient été trouvés au sein d'applications réelles, ce modèle a été conservé afin de fournir un modèle simplifié de la réalité. Il a, par exemple, été utilisé par Kouvelis *et al.* (2000) et Gourgand *et al.* (2005). Le seul exemple effectif envisagé est celui où la durée d'un traitement est dépendante du placement initial de la machine sur laquelle il est réalisé.

4.4.2.2.2 Loi exponentielle. Une durée de traitement p_{ij} est susceptible de suivre une loi exponentielle $\mathcal{E}(\lambda, a)$, où λ et a sont deux paramètres positifs (voir figure 4.10). Sa densité de probabilité est alors :

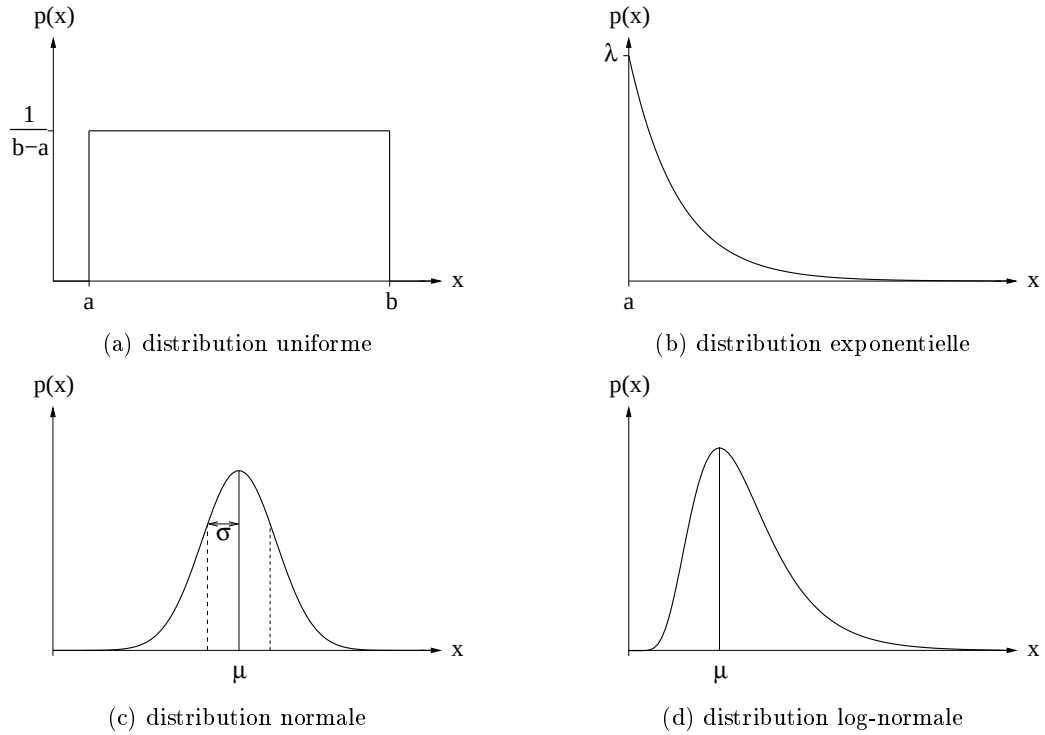
$$pdf(x) = \begin{cases} \lambda\, e^{-\lambda(x-a)} & \text{si } x \geq a \\ 0 & \text{sinon} \end{cases} \tag{4.13}$$

Les distributions exponentielles sont couramment utilisées pour modéliser les événements aléatoires qui risquent de se produire avec incertitude. C'est typiquement le cas en présence d'*aléas* ; c'est-à-dire lorsqu'une valeur est associée à un paramètre, mais que celle-ci est susceptible d'être modifiée par certains événements inattendus. Ce type d'incertitude est constaté dans les cas de panne, ou, plus généralement, dès lors qu'une machine nécessite une intervention humaine : réparation, entretien, recharge de composants nécessaires à son bon fonctionnement, réinstallation d'un objet mal placé, etc. La durée du traitement s'écoule alors jusqu'à la fin de l'intervention (ou jusqu'à la réparation de la machine endommagée). Plusieurs études dont les durée d'exécution ont été modélisées à l'aide d'une loi exponentielle ont été menées (Makino, 1965; Talwar, 1967; Cunningham et Dutta, 1973; Ku et Niu, 1986; Gourgand *et al.*, 2005).

4.4.2.2.3 Loi normale. Il s'avère également possible qu'une durée d'exécution p_{ij} suive une *loi normale* $\mathcal{N}(\mu, \sigma)$, où μ est l'espérance et $\sigma^2 > 0$ la variance de p_{ij} (voir figure 4.10). Dans ce cas, sa densité de probabilité est donnée par :

$$pdf(x) = \frac{1}{\sigma\sqrt{2\pi}}\, exp\Big(-\frac{1}{2}\Big(\frac{x-\mu}{\sigma}\Big)^2\Big) \tag{4.14}$$

Ce genre de cas est particulièrement fréquent en présence de facteurs humains. En effet, il semble peu probable qu'un ouvrier accomplisse une certaine tâche en un temps parfaitement identique à



(a) distribution uniforme

(b) distribution exponentielle

(c) distribution normale

(d) distribution log-normale

Fig. 4.10 – Exemples de fonctions de densité de probabilité susceptibles de modéliser une durée d'exécution aléatoire : (a) loi uniforme, (b) loi exponentielle, (c) loi normale (d) loi log-normale.

chaque exécution. Ce dernier la réalisera plus ou moins rapidement selon divers critères tels que son degré d'attention, sa forme physique, etc. Par ailleurs, il se peut qu'un traitement dépende de facteurs inconnus, incontrôlables ou décrits de manière imprécise ou ambiguë par l'analyste. Prenons l'exemple d'une usine chimique (Fortemps, 1997). Un traitement est, dans ce cas, une réaction chimique entre plusieurs composants. Or la durée d'une réaction chimique n'est pas fixe, elle peut varier selon la température, la pression, la qualité des composants, et d'autres paramètres dont certains peuvent être complètement inconnus. Il est donc impossible de vouloir contrôler toutes ces dépendances. Les durées d'exécutions observées varient alors selon une loi normale. Cette modélisation a, par exemple, été utilisée par Wang *et al.* (2005); Gourgand *et al.* (2005).

4.4.2.2.4 Loi log-normale. Une variable aléatoire x suit une *loi log-normale* de paramètres (μ, σ) si $\log x$ suit la loi $\mathcal{N}(\mu, \sigma)$ (voir figure 4.10). Sa densité de probabilité est alors :

$$pdf(x) = \begin{cases} \frac{1}{x\sigma\sqrt{2\pi}} \frac{1}{x} exp(-\frac{1}{2}(\frac{\log x - \mu}{\sigma})^2) & \text{si } x > 0 \\ 0 & \text{sinon} \end{cases} \tag{4.15}$$

La distribution log-normale est couramment utilisée pour modéliser l'influence de variables environnementales incontrôlables. Par exemple, cette modélisation a été utilisée par Maddox et Birge (1996) ainsi que Dauzère-Pérès *et al.* (2005).

4.4.2.2.5 Jeux de données. Les jeux de données que nous proposons étendent ceux qui ont été présentés au premier chapitre pour le problème de Flowshop multiobjectif [2]. Afin de générer de l'incertitude sur une instance déterministe initiale, les quatre distributions de probabilité proposées lors de la section précédente peuvent être appliquées sur les données initiales à l'aide d'un fichier de configuration. Nous avons choisi de permettre la configuration de cette incertitude sur les machines uniquement, en spécifiant, pour chacune d'elles, une distribution de probabilité et ses paramètres ou des proportions liées à sa tendance centrale. Ainsi, à chaque génération de l'incertitude sur une instance déterministe, les durées d'exécutions sont différentes. Nous obtenons alors une réalisation des paramètres environnementaux incertains, correspondant à un scénario, ceci afin de refléter au mieux la réalité du monde industriel.

4.4.2.3 Design expérimental

4.4.2.3.1 Scénarios. Au cours de nos expérimentations, nous considérons un ensemble de $(p+q)$ scénarios tout autant plausibles les uns que les autres : p scénarios pour la phase de recherche, et $q = 20$ scénarios pour la phase d'évaluation des performances. Deux valeurs de p sont considérées, $p = 10$ et $p = 20$. Celles-ci correspondent à un échantillon de vecteur objectif de taille 10 (respectivement 20) associé à chaque solution évaluée. À chaque scénario correspond une réalisation des paramètres environnementaux incertains, c'est-à-dire des durées d'exécutions stochastiques pour le cas du problème de Flowshop étudié ici. Pour créer ces différents scénarios, nous avons utilisé les modèles stochastiques définis précédemment. Ainsi, pour une instance donnée, nous avons créé p scénarios indépendants, dont les durées d'exécutions suivent des distributions uniforme, normale, exponentielle ou log-normale de la façon suivante :

- loi uniforme : $p_{ij} \sim \mathcal{U}(a = (1-\alpha) \times p_{ij},\, b = (1+\alpha) \times p_{ij})$;
- loi normale : $p_{ij} \sim \mathcal{N}(\mu = p_{ij},\, \sigma = \alpha \times p_{ij})$;
- loi exponentielle : $p_{ij} \sim \mathcal{E}(a = p_{ij},\, \lambda = \frac{1}{\alpha \times p_{ij}})$;
- loi log-normale : $p_{ij} \sim \mathit{log}\text{-}\mathcal{N}(\mu = \log p_{ij},\, \sigma = \alpha \times \log p_{ij})$;
- lois variables : la loi associée aux durées d'exécutions diffère sur chacune des machines.

Ainsi, la tendance centrale des lois correspond toujours à la durée d'exécution déterministe p_{ij} de l'instance considérée. Le paramètre α représente le degré de déviation des durées d'exécutions. Par la suite, nous allons tester deux valeurs : $\alpha = 0.10$ et $\alpha = 0.20$, ce qui correspond par exemple à une déviation de $\pm 10\%$ et $\pm 20\%$, respectivement, pour une loi uniforme.

4.4.2.3.2 Approches étudiées. Les algorithmes que nous allons considérer lors de nos expérimentations correspondent aux huit approches proposées intégrées au sein de l'algorithme IBEA. En plus, nous allons considérer une approche naïve qui se base sur un seul et unique scénario (le premier scénario considéré), notée z^1. La taille de l'échantillon associé à une solution est donc de 1, et l'algorithme résultant se comporte de la même façon que pour le cas déterministe. En plus, nous considérons également l'approche proposée par Basseur et Zitzler (2006), qui se base sur une estimation de l'espérance mathématique de la I-valeur associée à une solution pour le cas particulier de l'indicateur $I_{\epsilon+}$.

2. Les jeux de données pour le cas déterministe et le cas stochastique sont disponibles à l'URL : `http://www.lifl.fr/~liefooga/benchmarks/`.

Table 4.1 – Condition d'arrêt : nombre d'évaluations par exécution.

Instance	Nombre d'évaluations
$020 \times 05 \times 01$	500000
$020 \times 05 \times 02$	500000
$020 \times 10 \times 01$	1000000
$020 \times 10 \times 02$	1000000
$020 \times 20 \times 01$	2000000
$050 \times 05 \times 01$	5000000
$050 \times 10 \times 01$	10000000
$050 \times 20 \times 01$	20000000

4.4.2.3.3 Paramètres. Tout d'abord, la condition d'arrêt que nous allons utiliser se base sur un nombre d'évaluations maximum. Ainsi, plus la taille de l'échantillon associé à une solution est élevée, moins l'est le nombre d'itérations de l'algorithme, ceci dans le but d'évaluer l'importance de la taille de l'échantillon sur les performances des algorithmes.
La taille de la population a été fixée à 100 solutions. Les probabilités d'application de l'opérateur de croisement et de l'opérateur de mutation ont été respectivement fixées à 0.25 et 1.0.

4.4.2.4 Résultats et discussion

Pour des raisons synthétiques, seuls les résultats dont les lois associées aux durées d'exécutions varient d'une machine à l'autre seront présentés. En outre, pour les deux protocoles considérés, les approches concernées par la prise en compte de l'incertitude dans le meilleur des cas, le pire des cas, le cas moyen et le cas meilleur sont respectivement comparées les unes aux autres par rapport au protocole d'évaluation correspondant. Par exemple, la comparaison des résultats obtenus vis-à-vis du vecteur objectif moyen se base uniquement sur les approches z^{avg} et I^{avg}, avec pour chacune d'elles deux paramétrages différentes : un échantillon de 10 vecteurs objectifs par solution, et un échantillon de 20 vecteurs objectifs par solution.
Tout d'abord, par rapport à nos stratégies, l'approche proposée par Basseur et Zitzler (2006) se base sur une estimation de l'espérance mathématique de la I-valeur associée à une solution. En pratique, cette estimation s'est avérée très coûteuse en temps de calcul, et l'algorithme correspondant s'est montré largement plus gourmand que les approches que nous avons proposées. Par ailleurs, les performances qu'elle a obtenues ne rivalisaient généralement pas avec les autres méthodes, du moins pour les protocoles d'évaluation que nous avons utilisés. Mais il est vrai que les auteurs ont reporté de meilleurs résultats pour des problèmes à trois objectifs dans leur étude sur des fonctions multiobjectif continues (Basseur et Zitzler, 2006). Or nous nous sommes pour l'instant limités à une étude sur le problème de Flowshop à deux objectifs.
Les résultats obtenus selon le protocole d'évaluation des performances basé sur une évaluation représentative sont présentés dans les tableaux 4.2, 4.3, 4.4 et 4.5. Et les résultats obtenus selon le protocole d'évaluation des performances basé sur les valeurs d'indicateur sont présentés dans les tableaux 4.6, 4.7, 4.8 et 4.9. Pour chacun des cas sont respectivement représentés le meilleur des cas, le pire des cas, le cas moyen, et le cas médian.

Une première remarque cruciale que nous pouvons formuler est que les méthodes prenant en compte l'incertitude se sont avérées globalement plus performantes que la méthode z^1, pour laquelle un scénario unique est considéré, et ceci pour la quasi-totalité des instances. Les seuls

TABLE 4.2 – Comparaison des algorithmes selon le vecteur objectif dans le meilleur des cas.

	$\alpha = 0.10$										$\alpha = 0.20$									
	I_H^-					$I_{\epsilon+}$					I_H^-					$I_{\epsilon+}$				
	z^1	z^{best}		I^{best}		z^1	z^{best}		I^{best}		z^1	z^{best}		I^{best}		z^1	z^{best}		I^{best}	
		10	20	10	20		10	20	10	20		10	20	10	20		10	20	10	20
$020 \times 05 \times 01$	**0**	1	3	1	3	**0**	1	3	1	2	3	**0**	1	2	3	2	**0**	**0**	3	4
$020 \times 05 \times 02$	4	**0**	1	**0**	2	4	**0**	**0**	**0**	**0**	4	**0**	**0**	1	3	4	**0**	**0**	2	3
$020 \times 10 \times 01$	1	1	1	**0**	1	1	1	1	**0**	1	4	**0**	2	**0**	2	3	**0**	2	**0**	2
$020 \times 10 \times 02$	**0**	1	1	2	3	**0**	1	1	1	2	2	**0**	**0**	2	2	2	**0**	**0**	3	4
$020 \times 20 \times 01$	4	1	1	**0**	1	4	1	1	**0**	1	1	**0**	**0**	**0**	2	1	**0**	**0**	1	3
$050 \times 05 \times 01$	**0**	2	2	1	2	**0**	2	2	**0**	2	**0**	2	1	1	1	**0**	1	1	1	1
$050 \times 10 \times 01$	**0**	1	4	**0**	**0**	**0**	1	4	**0**	**0**	1	**0**	**0**	**0**	**0**	1	**0**	1	**0**	3
$050 \times 20 \times 01$	2	3	3	**0**	**0**	2	3	3	**0**	**0**	**0**	**0**	2	1	2	**0**	1	1	1	3

TABLE 4.3 – Comparaison des algorithmes selon le vecteur objectif dans le pire des cas.

	$\alpha = 0.10$										$\alpha = 0.20$									
	I_H^-					$I_{\epsilon+}$					I_H^-					$I_{\epsilon+}$				
	z^1	z^{worst}		I^{worst}		z^1	z^{worst}		I^{worst}		z^1	z^{worst}		I^{worst}		z^1	z^{worst}		I^{worst}	
		10	20	10	20		10	20	10	20		10	20	10	20		10	20	10	20
$020 \times 05 \times 01$	4	**0**	2	**0**	2	4	**0**	2	**0**	2	4	3	**0**	**0**	1	4	2	**0**	**0**	**0**
$020 \times 05 \times 02$	4	**0**	**0**	**0**	1	4	**0**	**0**	**0**	**0**	4	1	3	**0**	1	4	2	3	1	**0**
$020 \times 10 \times 01$	2	**0**	**0**	1	3	1	**0**	**0**	3	3	3	**0**	**0**	2	**0**	4	1	**0**	2	**0**
$020 \times 10 \times 02$	4	1	**0**	**0**	1	4	1	2	**0**	**0**	**0**	3	2	1	1	**0**	3	2	**0**	**0**
$020 \times 20 \times 01$	4	2	1	**0**	**0**	4	2	**0**	**0**	**0**	4	2	1	**0**	1	4	1	1	**0**	1
$050 \times 05 \times 01$	3	**0**	**0**	**0**	1	2	**0**	**0**	2	2	3	3	1	2	**0**	3	3	1	1	**0**
$050 \times 10 \times 01$	4	**0**	1	**0**	**0**	4	**0**	1	**0**	**0**	4	2	3	**0**	1	4	2	3	**0**	1
$050 \times 20 \times 01$	4	1	**0**	**0**	**0**	4	2	1	1	**0**	4	3	**0**	1	**0**	4	3	**0**	**0**	**0**

TABLE 4.4 – Comparaison des algorithmes selon le vecteur objectif dans le cas moyen.

	$\alpha = 0.10$										$\alpha = 0.20$									
	I_H^-					$I_{\epsilon+}$					I_H^-					$I_{\epsilon+}$				
	z^1	z^{avg}		I^{avg}		z^1	z^{avg}		I^{avg}		z^1	z^{avg}		I^{avg}		z^1	z^{avg}		I^{avg}	
		10	20	10	20		10	20	10	20		10	20	10	20		10	20	10	20
$020 \times 05 \times 01$	1	**0**	3	**0**	3	1	**0**	3	**0**	3	4	**0**	**0**	**0**	1	4	**0**	**0**	**0**	2
$020 \times 05 \times 02$	3	**0**	1	2	2	2	**0**	**0**	2	2	4	**0**	**0**	**0**	3	4	**0**	**0**	**0**	3
$020 \times 10 \times 01$	1	**0**	1	1	3	1	**0**	1	3	3	4	**0**	**0**	3	2	3	**0**	**0**	3	2
$020 \times 10 \times 02$	**0**	**0**	**0**	**0**	2	1	**0**	**0**	2	3	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
$020 \times 20 \times 01$	4	**0**	**0**	**0**	3	4	**0**	**0**	**0**	3	4	1	**0**	2	**0**	4	1	**0**	1	**0**
$050 \times 05 \times 01$	2	**0**	**0**	3	3	1	**0**	**0**	3	3	4	**0**	1	2	3	3	**0**	1	2	3
$050 \times 10 \times 01$	3	**0**	1	1	3	3	**0**	1	2	3	4	**0**	**0**	3	**0**	4	**0**	**0**	3	2
$050 \times 20 \times 01$	4	**0**	**0**	2	1	4	**0**	**0**	2	2	4	1	**0**	3	1	4	1	**0**	1	1

TABLE 4.5 – Comparaison des algorithmes selon le vecteur objectif dans le cas médian.

	$\alpha = 0.10$										$\alpha = 0.20$									
	I_H^-					$I_{\epsilon+}$					I_H^-					$I_{\epsilon+}$				
	z^1	z^{med}		I^{med}		z^1	z^{med}		I^{med}		z^1	z^{med}		I^{med}		z^1	z^{med}		I^{med}	
		10	20	10	20		10	20	10	20		10	20	10	20		10	20	10	20
$020 \times 05 \times 01$	**0**	1	4	**0**	3	**0**	**0**	3	**0**	3	4	**0**	1	**0**	1	4	**0**	**0**	**0**	**0**
$020 \times 05 \times 02$	4	**0**	2	**0**	**0**	4	**0**	**0**	**0**	**0**	4	**0**	**0**	**0**	**0**	4	**0**	1	**0**	**0**
$020 \times 10 \times 01$	2	**0**	1	**0**	2	**0**	**0**	1	**0**	4	4	2	**0**	2	**0**	3	1	**0**	3	**0**
$020 \times 10 \times 02$	1	**0**	1	**0**	1	1	**0**	1	**0**	1	3	4	**0**	**0**	**0**	3	3	**0**	**0**	**0**
$020 \times 20 \times 01$	4	1	1	**0**	**0**	4	1	**0**	**0**	**0**	4	1	**0**	1	**0**	3	**0**	**0**	3	**0**
$050 \times 05 \times 01$	3	**0**	3	**0**	**0**	**0**	**0**	**0**	**0**	**0**	4	**0**	**0**	1	**0**	2	**0**	**0**	2	2
$050 \times 10 \times 01$	4	**0**	**0**	**0**	**0**	4	**0**	**0**	**0**	**0**	4	2	2	**0**	**0**	4	3	2	**0**	**0**
$050 \times 20 \times 01$	4	2	**0**	1	**0**	3	2	**0**	1	**0**	4	**0**	2	**0**	**0**	4	**0**	2	**0**	**0**

TABLE 4.6 – Comparaison des algorithmes selon la I-valeur dans le meilleur des cas.

	$\alpha = 0.10$										$\alpha = 0.20$									
	I_H^-					$\mathrm{I}_{\epsilon+}$					I_H^-					$\mathrm{I}_{\epsilon+}$				
	z^1	z^{best}		I^{best}		z^1	z^{best}		I^{best}		z^1	z^{best}		I^{best}		z^1	z^{best}		I^{best}	
		10	20	10	20		10	20	10	20		10	20	10	20		10	20	10	20
$020 \times 05 \times 01$	**0**	2	3	1	3	**0**	1	3	1	3	**0**	**0**	2	**0**	**0**	**0**	**0**	1	**0**	**0**
$020 \times 05 \times 02$	**0**	**0**	1	**0**	4	3	**0**	1	**0**	3	4	**0**	**0**	1	3	4	**0**	**0**	2	3
$020 \times 10 \times 01$	**0**	1	1	**0**	2	**0**	1	1	**0**	1	1	**0**	**0**	2	1	2	**0**	1	1	2
$020 \times 10 \times 02$	**0**	1	**0**	3	4	**0**	1	**0**	3	4	**0**	**0**	**0**	2	**0**	**0**	3	1	**0**	**0**
$020 \times 20 \times 01$	2	2	**0**	**0**	**0**	2	2	1	**0**	1	1	**0**	1	1	1	1	**0**	**0**	2	1
$050 \times 05 \times 01$	**0**	3	3	**0**	2	**0**	2	2	1	2	**0**	3	2	**0**	**0**	**0**	3	3	**0**	**0**
$050 \times 10 \times 01$	1	2	4	**0**	**0**	**0**	1	4	**0**	**0**	4	2	2	**0**	**0**	3	1	1	**0**	**0**
$050 \times 20 \times 01$	2	3	3	**0**	**0**	2	3	3	**0**	**0**	**0**	1	1	**0**	1	**0**	**0**	1	**0**	**0**

TABLE 4.7 – Comparaison des algorithmes selon la I-valeur dans le pire des cas.

	$\alpha = 0.10$										$\alpha = 0.20$									
	I_H^-					$\mathrm{I}_{\epsilon+}$					I_H^-					$\mathrm{I}_{\epsilon+}$				
	z^1	z^{worst}		I^{worst}		z^1	z^{worst}		I^{worst}		z^1	z^{worst}		I^{worst}		z^1	z^{worst}		I^{worst}	
		10	20	10	20		10	20	10	20		10	20	10	20		10	20	10	20
$020 \times 05 \times 01$	4	**0**	2	**0**	**0**	4	**0**	**0**	**0**	**0**	4	3	1	**0**	**0**	4	2	2	**0**	**0**
$020 \times 05 \times 02$	3	2	2	**0**	**0**	2	2	1	**0**	**0**	4	2	2	**0**	1	4	2	2	**0**	**0**
$020 \times 10 \times 01$	3	**0**	1	2	2	1	**0**	**0**	2	2	3	3	1	1	**0**	2	2	1	2	**0**
$020 \times 10 \times 02$	2	2	3	**0**	**0**	3	**0**	2	**0**	**0**	2	2	2	**0**	**0**	2	2	2	**0**	**0**
$020 \times 20 \times 01$	4	1	1	**0**	**0**	4	**0**	**0**	**0**	**0**	2	2	2	**0**	**0**	2	2	2	1	**0**
$050 \times 05 \times 01$	1	**0**	2	**0**	2	**0**	**0**	**0**	1	3	3	2	2	**0**	1	2	2	2	**0**	**0**
$050 \times 10 \times 01$	4	1	2	**0**	1	4	**0**	**0**	**0**	2	3	2	2	**0**	1	4	2	2	**0**	**0**
$050 \times 20 \times 01$	4	2	2	**0**	**0**	4	1	1	1	**0**	4	2	2	**0**	**0**	3	2	2	**0**	**0**

TABLE 4.8 – Comparaison des algorithmes selon la I-valeur dans le cas moyen.

	$\alpha = 0.10$										$\alpha = 0.20$									
	I_H^-					$\mathrm{I}_{\epsilon+}$					I_H^-					$\mathrm{I}_{\epsilon+}$				
	z^1	z^{avg}		I^{avg}		z^1	z^{avg}		I^{avg}		z^1	z^{avg}		I^{avg}		z^1	z^{avg}		I^{avg}	
		10	20	10	20		10	20	10	20		10	20	10	20		10	20	10	20
$020 \times 05 \times 01$	2	**0**	2	**0**	2	4	**0**	2	**0**	2	4	**0**	**0**	**0**	**0**	4	**0**	**0**	**0**	**0**
$020 \times 05 \times 02$	4	**0**	**0**	**0**	1	4	**0**	**0**	2	**0**	4	2	1	**0**	**0**	4	2	1	**0**	**0**
$020 \times 10 \times 01$	2	**0**	1	2	2	2	**0**	**0**	3	2	4	2	**0**	3	1	3	1	**0**	3	1
$020 \times 10 \times 02$	2	**0**	**0**	**0**	2	1	**0**	**0**	1	2	4	1	2	**0**	**0**	4	1	2	**0**	**0**
$020 \times 20 \times 01$	4	**0**	**0**	**0**	**0**	4	**0**	**0**	**0**	2	4	3	1	2	**0**	4	2	1	2	**0**
$050 \times 05 \times 01$	2	**0**	1	2	3	2	**0**	**0**	3	3	4	**0**	1	1	1	4	**0**	1	2	2
$050 \times 10 \times 01$	4	**0**	1	1	3	3	**0**	1	2	3	4	1	1	**0**	**0**	4	1	**0**	**0**	**0**
$050 \times 20 \times 01$	4	1	**0**	3	1	4	1	**0**	3	2	4	3	1	1	**0**	4	3	1	1	**0**

TABLE 4.9 – Comparaison des algorithmes selon la I-valeur dans le cas médian.

	$\alpha = 0.10$										$\alpha = 0.20$									
	I_H^-					$\mathrm{I}_{\epsilon+}$					I_H^-					$\mathrm{I}_{\epsilon+}$				
	z^1	z^{med}		I^{med}		z^1	z^{med}		I^{med}		z^1	z^{med}		I^{med}		z^1	z^{med}		I^{med}	
		10	20	10	20		10	20	10	20		10	20	10	20		10	20	10	20
$020 \times 05 \times 01$	1	1	4	**0**	3	1	**0**	3	**0**	3	4	**0**	2	**0**	**0**	4	1	1	**0**	**0**
$020 \times 05 \times 02$	4	**0**	**0**	**0**	**0**	3	2	**0**	2	**0**	4	**0**	**0**	**0**	**0**	4	2	**0**	**0**	**0**
$020 \times 10 \times 01$	4	**0**	**0**	**0**	2	3	**0**	**0**	**0**	3	4	2	**0**	3	1	4	1	**0**	3	1
$020 \times 10 \times 02$	2	1	1	**0**	1	2	**0**	2	**0**	1	3	3	**0**	**0**	**0**	3	3	**0**	**0**	**0**
$020 \times 20 \times 01$	4	2	2	**0**	**0**	4	1	**0**	**0**	**0**	4	3	1	1	**0**	4	2	**0**	2	**0**
$050 \times 05 \times 01$	3	**0**	3	**0**	1	1	**0**	1	1	1	4	**0**	**0**	2	**0**	4	**0**	**0**	2	**0**
$050 \times 10 \times 01$	4	**0**	**0**	**0**	**0**	3	**0**	**0**	**0**	**0**	4	2	2	1	**0**	4	3	2	1	**0**
$050 \times 20 \times 01$	4	2	**0**	2	**0**	3	1	**0**	3	**0**	4	1	3	**0**	**0**	4	1	2	**0**	**0**

cas où z^1 a obtenu de meilleurs résultats que les autres approches apparaissent lors de la prise en compte du meilleur des cas, ceci pour les deux protocoles d'évaluation des performances que nous avons considéré. Preuve que les approches déterministes ne peuvent rivaliser avec les techniques stochastiques, même les plus basiques.
Quant au premier protocole d'évaluation, basé sur des évaluations représentatives, les résultats sont assez variés d'un type de préférence à l'autre, et d'un degré de déviations à l'autre (α). Nous pouvons tout de même remarquer les très bonnes performances de l'approche basée sur le vecteur objectif moyen lors de la considération du cas moyen. Globalement, pour une faible incertitude ($\alpha = 10$), les approches basées sur les I-valeurs semblent plus performantes, alors qu'aucune conclusion claire ne peut être dégagée pour un taux d'incertitude plus élevé pour ce protocole d'évaluation (le cas moyen mis à part).
Concernant le protocole d'évaluation basé sur les I-valeurs, les deux classes de méthodes se comportent relativement aussi bien l'une que l'autre en présence d'incertitude limitée ($\alpha = 10$), même si les approches basées sur les approches basées sur un vecteur objectif représentatif semblent légèrement plus performantes. Au contraire, dès que l'incertitude augmente ($\alpha = 20$), il apparaît clairement que les approches basées sur les I-valeurs sont plus performantes.
Enfin, à propos de l'influence de la taille de l'échantillon associé à une solution, il semble que celle-ci soit corrélée au degré d'incertitude considéré. En effet, dans bien des cas, pour une même approche de résolution, un échantillon de taille 10 s'avère généralement plus efficace pour un degré de déviation de 10%, alors qu'un échantillon de taille 20 est plus efficace pour un degré de déviation de 20%.

Conclusion

Ce chapitre traite de la modélisation et de la résolution de problèmes d'optimisation multiobjectif en environnement incertain. L'intérêt de la définition, de l'étude et de la résolution de ce type de problème a été exposé, aussi bien d'un point de vue académique que d'un point de vue pratique et industriel. En effet, nous avons soutenu que bien des problèmes réalistes impliquaient une certaine forme d'incertitude. Cette incertitude peut provenir de différentes sources, comme des fonctions objectif en elles-mêmes, des variables de décision ou des paramètres environnementaux. Aussi, avec un effort raisonnable, les métaheuristiques peuvent être efficacement appliquées à la résolution de tels problèmes, et ceci pour des problèmes dont les formulations stochastiques sont très différentes les unes des autres. Les contributions majeures de ce chapitre sont résumées ci-dessous.

Classification. Un point complet des métaheuristiques proposées jusqu'ici pour la résolution de problèmes d'optimisation multiobjectif stochastiques a été réalisé. Dès lors que la prise en compte de l'incertitude se base sur un échantillonnage, nous avons affirmé que la comparaison de solutions deux à deux se traduisait par la comparaison d'ensembles de vecteurs objectif. Aussi, un travail d'abstraction à propos de la conception de telles méthodes, et une classification des approches existantes, basée sur le choix et l'exploitation de valeurs représentatives, ont été proposés.

Nouvelles approches métaheuristiques. Nous avons également proposé de nouvelles approches méthodologiques dédiées à la résolution très générale de problèmes d'optimisation multiobjectif sous incertitude. Celles-ci se basent sur l'ajout d'un niveau supplémentaire au

sein des stratégies d'affectation des valeurs de fitness basées sur un indicateur binaire de qualité. Elles étendent donc l'approche proposée par Zitzler et Künzli (2004) au cas stochastique en interprétant de diverses façons le degré d'aversion à l'incertitude du décideur ou du praticien. Pour cela, elles se basent sur la prise en compte du meilleur des cas, du pire des cas, du cas moyen et du cas médian.

Évaluation des performances. Nous avons ensuite discuté de la question de l'évaluation des performances dans le cadre de l'optimisation multiobjectif en environnement incertain. Nous avons vu que cette notion, qui n'est déjà pas une tâche facile dans le cas déterministe, était encore plus délicate pour le cas stochastique. Nous avons fourni divers éléments de réponse, correspondant là encore à différents degrés de conservatisme quant à l'incertitude.

Résolution du problème de Flowshop à durées d'exécutions stochastiques. Une formulation stochastique du problème de Flowshop a pour la première fois été considérée dans un contexte multiobjectif. Suite à une étude des différentes sources d'incertitude possibles, nous avons retenu les durées d'exécutions que nous avons modélisées à l'aide de variables aléatoires. En ce sens, différents jeux de données ont d'abord été proposés. Puis, nous avons appliqué et expérimenté les méthodologies proposées dans ce chapitre, aussi bien en termes de méthodes de résolution que d'évaluation des performances, à ce problème de Flowshop stochastique. Les expérimentations ont révélé que la prise en compte de l'incertitude se montrait indispensable pour obtenir un ensemble de solutions robustes aux paramètres environnementaux incertains.

Bien qu'encore relativement naissant dans le cadre de l'optimisation multiobjectif, la prise en compte de l'incertitude dans la formulation du problème et dans les approches de résolution métaheuristiques connaît un intérêt grandissant, de par son importance pratique incontestable. Dans ce chapitre, nous espérons avoir soulevé l'intérêt de ce domaine de recherche, ainsi que ses répercussions engendrées sur les méthodologies de résolution, en particulier pour les métaheuristiques. Nous pensons en effet que ces dernières ont indéniablement un rôle à jouer pour la résolution de tels problèmes, même si beaucoup de questions fondamentales restent encore ouvertes.

Conclusion générale

Les travaux présentés dans ce mémoire de thèse traitent de la conception, de l'implémentation et de l'analyse expérimentale d'approches métaheuristiques pour l'optimisation multiobjectif. En particulier, l'intérêt des approches coopératives et de la prise en compte de l'incertitude a été montré dans le cadre de problématiques liées à la logistique. Le but de cette dernière partie est de synthétiser tout d'abord les principales contributions que nous avons exposées, puis de discuter de plusieurs directions ouvertes pour de futurs travaux de recherche.

Synthèse des contributions

Métaheuristiques. Au cours de la première partie, nous avons proposé une vue unifiée de la conception de métaheuristiques pour l'optimisation multiobjectif. Nous avons identifié les concepts communs partagés par un grand nombre d'approches, en délimitant les composants spécifiques au problème traité de la partie invariante impliquée dans cette classe de méthodes de résolution. Nous avons montré qu'en plus des composants classiques des métaheuristiques pour l'optimisation monoobjectif, les métaheuristiques pour l'optimisation multiobjectif reposent sur trois composants de recherche principaux qui leurs sont propres : l'affectation des valeurs de fitness, la préservation de la diversité, et l'élitisme. Partant de cette unification, nous nous sommes particulièrement intéressés à deux méthodologies (les algorithmes évolutionnaires et les algorithmes de recherche locale), pour lesquels nous avons proposé des modèles de conception. Nous avons illustré leur robustesse et leur fiabilité en traitant un certain nombre d'algorithmes classiques comme de simples instances. Ces modèles nous ont d'ailleurs permis de développer de nouvelles métaheuristiques pour l'optimisation multiobjectif, à savoir un algorithme évolutionnaire élitiste (SEEA), et diverses stratégies de recherche locale basées sur une relation de dominance (méthodes DMLS). Ces dernières présentent une alternative intéressante aux algorithmes évolutionnaires, présents en très grand nombre au sein de la littérature.
Cette étude approfondie à propos de la conception de métaheuristiques pour l'optimisation multiobjectif a débouché sur la proposition d'une plateforme logicielle dédiée à l'implémentation de tels algorithmes : ParadisEO-MOEO. Celle-ci met à disposition les composants de recherche les plus répandus de l'optimisation multiobjectif. Elle fournit un haut degré de flexibilité et de polyvalence, puisque le praticien, le chercheur ou le décideur peuvent aisément modifier ou ajouter de nouveaux composants, et aussi résoudre de nouveaux problèmes. De plus, les métaheuristiques les plus populaires (par exemple NSGA-II, SPEA2, IBEA ou PLS) sont disponibles au sein de la plateforme ; leur conception et leur code sont donc directement réutilisables. Une telle plateforme permet de s'abstraire de l'implémentation parfois complexe des mécanismes de résolution, et de se concentrer sur les aspects spécifiques au problème étudié. L'utilisation de ParadisEO permet par ailleurs de réutiliser les composants de recherche développés pour un problème monoobjectif, mais également d'implémenter, de façon incrémentale, des métaheuristiques hybrides et parallèles pour l'optimisation multiobjectif. Ceci fait de ParadisEO-MOEO une plateforme logicielle unique, pour grande part responsable du succès de ParadisEO [3].

3. Depuis près de 3 ans que la ParadisEO est hébergée sur le site `http://paradiseo.gforge.inria.fr`, plus de 9000 téléchargements et plus de 200 utilisateurs abonnés à sa liste de diffusion ont été comptabilisés.

Enfin, nous avons analysé expérimentalement les performances de plusieurs métaheuristiques appliquées à la résolution de deux problèmes d'optimisation combinatoire que nous avons étudiés tout au long de ce manuscrit, les problèmes académiques de Flowshop et de Ring-Star. Ces deux problèmes illustrent bien ce que l'on peut typiquement trouver dans l'industrie en optimisation multiobjectif. Premièrement, cette analyse expérimentale nous a permis de mettre en évidence plusieurs lignes directrices utiles à propos du choix des principaux composants de recherche des algorithmes DMLS. Deuxièmement, nous avons pu résoudre le problème de Ring-Star, pour la première fois comme un problème d'optimisation multiobjectif ; problème pour lequel nos métaheuristiques ont obtenu des résultats encourageants vis-à-vis des méthodes exactes existantes pour le problème de Ring-Star monoobjectif.

Métaheuristiques coopératives. En second lieu, nous nous sommes intéressés à l'élaboration de métaheuristiques hybrides pour l'optimisation multiobjectif. Dans ce contexte, nous avons tout d'abord recensé les différents types de coopération rencontrés dans la littérature, en particulier dans le cadre de l'optimisation multiobjectif. Deux nouvelles approches coopératives, hybridant toutes deux des métaheuristiques de haut niveau, ont ensuite été proposées.
La première approche coopérative se base sur une hybridation en mode relais, où un algorithme évolutionnaire et un algorithme de recherche locale sont exécutés séquentiellement. L'originalité de ce schéma d'hybridation réside dans le fait que les étapes de transition d'une méthode à l'autre sont répétées à plusieurs reprises, au contraire des hybridations existantes qui se contentent généralement de les exécuter l'une à la suite de l'autre, avant de stopper purement et simplement le processus de recherche. Une autre particularité est qu'aucune technique d'agrégation des différentes fonctions objectif n'est ici considérée lors de la phase de recherche locale. Par ailleurs, deux variantes de ce même modèle de coopération ont été développées. La première opère, à chaque étape de l'algorithme principal, une transition systématique depuis l'algorithme évolutionnaire vers l'algorithme de recherche locale. Au sein de la deuxième variante, cette même transition a lieu de façon adaptative, uniquement si une condition liée à la convergence de l'algorithme est vérifiée. Au cours de nos expérimentations, nous avons montré l'intérêt de ce type d'hybridation, même si la condition de transition adaptative ne s'est pas révélée plus performante que l'approche plus naïve, probablement en raison du temps de calcul plus élevé qu'elle nécessite.
Le deuxième modèle d'hybridation proposé repose sur une division uniforme de l'espace objectif à l'aide de points de référence multiples. Un nombre d'algorithmes évolutionnaires équivalent au nombre de points de référence définis est alors exécuté en parallèle. Chacun d'entre eux a pour but de trouver un sous-ensemble de solutions non-dominées, délimité par le point de référence qui lui est associé, et donc par la sous-région de l'espace objectif qui lui est impartie. Aussi, cette phase se prête naturellement aux calculs parallèles. Les résultats sont finalement combinés les uns aux autres afin de former une approximation complète de l'ensemble Pareto optimal. Les expérimentations ont été menées dans un environnement de calcul distribué, et ont révélé un comportement significativement plus performant qu'un ensemble de métaheuristiques non hybrides et séquentielles équivalentes.

Métaheuristiques en environnement incertain. Pour finir, nous avons vu que beaucoup de problèmes d'optimisation sont caractérisés par la présence d'incertitude. Cette incertitude peut provenir de différentes sources, comme des fonctions objectif en elles-mêmes, des variables de décision ou des paramètres environnementaux. Or, même si l'optimisation sous incertitude

est raisonnablement étudiée dans le cas monoobjectif, ce domaine de recherche est très limité en multiobjectif. L'état de l'art et la classification des approches que nous avons réalisés sur le sujet montrent que, bien que limité pour le moment, l'intérêt grandissant porté à ce type de problème est de plus en plus significatif.
Dès que nous traitons l'incertitude à l'aide d'un échantillonnage, nous avons mis en avant que le concept de base se traduisait par la comparaison d'ensembles, depuis l'espace décisionnel vers l'espace objectif. Pour cela, différentes approches de résolution ont été proposées. Un certain nombre d'entre elles se basent sur la transformation de l'ensemble de vecteurs objectif associé à une solution en un point unique. D'autres opèrent cette transformation à un niveau plus fin, ceci au sein d'une stratégie d'affectation des valeurs de fitness basée sur un indicateur de qualité. Pour chacune de ces deux classes, diverses techniques proposent de prendre position en fonction du meilleur des cas, du pire des cas, du cas moyen et du cas médian. Par ailleurs, nous avons discuté de la question essentielle liée à l'évaluation des performances dans le cadre de l'optimisation multiobjectif en environnement incertain. Ce n'est déjà pas une tâche facile en optimisation multiobjectif classique, et il n'existe pas encore de solutions probantes dans le cas incertain. Nous avons tenté de fournir quelques éléments de réponse, même si un effort reste à fournir dans ce domaine. Enfin, tous les concepts et méthodologies introduits pour la prise en compte de l'incertitude en optimisation multiobjectif ont été illustrés et mis en pratique sur le problème de Flowshop multiobjectif à durées d'exécutions stochastiques. Les expérimentations que nous avons menées ont montré qu'il s'avérait indispensable de tenir compte de l'incertitude au sein des mécanismes de résolution afin de traiter au mieux de problèmes pour lesquels les paramètres environnementaux sont stochastiques.

Perspectives

Bien évidemment, le travail présenté au sein de ce manuscrit de thèse ne se veut pas définitif. Nous espérons qu'il a permis de soulever beaucoup de perspectives ouvertes à investigation. Quelques idées et orientations possibles pour de futures recherches sont mentionnées ci-dessous.

Applications à d'autres problèmes d'optimisation. Tout d'abord, bien que l'ensemble des concepts et des principes introduits dans cette thèse soit potentiellement applicable à tout type de problème d'optimisation multiobjectif, il semble indéniable qu'ils se doivent d'être appliqués et expérimentés sur d'autres problèmes. Ceci permettrait de valider davantage les approches proposées. Pour cela, il pourrait notamment se montrer intéressant de considérer des problèmes à plus de deux ou trois fonctions objectif, auxquels nous nous sommes restreints jusqu'ici. Il pourrait par ailleurs être envisagé de traiter des applications plus concrètes, issues du monde réel. Ce type de problème est souvent plus difficile à résoudre que les problèmes académiques, du fait de la taille de leurs données, de leurs nombreuses contraintes, et du temps de calcul fréquemment limité qui est imparti aux méthodes employées pour leur résolution.

Problème de Ring-Star. Concernant le problème de Ring-Star, une première extension serait d'étendre la formulation du problème biobjectif proposée en une formulation à trois objectifs. En effet, comme l'ont proposé Beasley et Nascimento (1996), et plus tard Vogt *et al.* (2007) dans le cadre d'un problème monobjectif, les nœuds non visités pourraient soit être affectés au cycle, soit

être isolés ; auquel cas la fonction objectif additionnelle concernerait la minimisation du nombre de nœuds isolés.
Par ailleurs, bien que les approches que nous ayons proposées pour la résolution du problème de Ring-Star soient déjà prometteuses, plusieurs axes de recherche restent ouverts. Tout d'abord, nous pensons qu'il est possible d'améliorer la stratégie d'initialisation de la population utilisée au sein de chacune des métaheuristiques proposées. Deuxièmement, nous avons souligné que l'opérateur de croisement que nous avons adopté avait tendance à briser la structure du cycle des individus Parent chez les individus Enfant. Ainsi, comme l'ont proposé Renaud *et al.* (2004), nous pourrions employer une heuristique, voire une méthode exacte, dédiée à la résolution du problème du voyageur de commerce (TSP) afin d'améliorer le coût de l'anneau des solutions générées. Au vu de la littérature abondante concernant la résolution du TSP, il existe très certainement des méthodes de résolution très performantes qui pourraient nous aider à améliorer nos méthodes. Enfin, étant donné le nombre d'opérateurs de mutation intervenant au sein des algorithmes évolutionnaires, il serait intéressant de déterminer des taux appropriés pour chacun d'entre eux de façon adaptative, comme défini par Basseur (2005). Ces deux derniers points pourraient très largement améliorer l'efficacité des algorithmes évolutionnaires, et seraient donc également bénéfiques pour les méthodes coopératives.
Au sein d'une étude préliminaire menée sur le sujet (Liefooghe *et al.*, 2008c), nous avons déjà abordé les questions de l'initialisation de la population, et de l'intégration d'une heuristique monoobjectif élaborée pour le TSP, la méthode GENIUS (Gendreau *et al.*, 1992). Les expérimentations montrent que l'utilisation de ce schéma d'hybridation permet une amélioration significative des ensembles de solutions non-dominées, en particulier vis-à-vis de la fonction objectif relative au coût de l'anneau.

Unification de métaheuristiques pour l'optimisation multiobjectif. Concernant la vue unifiée de métaheuristiques pour l'optimisation multiobjectif, nous nous sommes pour l'instant limités à une étude approfondie de deux sous-classes de méthodologies : les algorithmes évolutionnaires et les algorithmes de recherche locale. Néanmoins, nous sommes certains qu'un grand nombre de composants impliqués au sein des modèles de conception présentés sont partagés par d'autres métaheuristiques. Par la suite, nous prévoyons de généraliser le modèle d'unification proposé à d'autres approches existantes. Ainsi, les algorithmes par essaims particulaires ou les algorithmes de *Scatter Search* (Talbi, 2009), qui semblent bien adaptés à l'optimisation multiobjectif, pourraient se montrer intéressants à approfondir dans le but d'étudier leur modularité et leur capacité à fournir un modèle de conception flexible.

ParadisEO-MOEO. En lien avec le point précédent, il est certain que l'addition de nouvelles métaheuristiques serait un plus pour la popularité et le développement de ParadisEO-MOEO. Pourtant, plutôt que de multiplier les propositions méthodologiques, il serait dans un premier temps nécessaire d'approfondir les notions déjà existantes au sein de la plateforme. Ainsi, plusieurs algorithmes évolutionnaires populaires, et leurs mécanismes particuliers découpés finement, seront bientôt intégrés. Nous pensons en particulier aux algorithmes PESA (Corne *et al.*, 2000), ϵ-MOEA (Deb *et al.*, 2005a), ou encore à différentes approches basées sur l'indicateur hypervolume (Emmerich *et al.*, 2005; Beume *et al.*, 2007; Bader et Zitzler, 2008). De surcroît, un effort reste à fournir concernant la facilité d'utilisation des méthodes hybrides et parallèles dédiées à l'optimisation multiobjectif. Nous pensons ensuite qu'il pourrait se montrer profitable d'élaborer

des stratégies d'archivage rapides des solutions non-dominées trouvées en cours de recherche par les métaheuristiques. En effet, lors de nos diverses expérimentations, il s'est avéré que les étapes de mise à jour de l'archive étaient bien souvent les phases les plus coûteuses de la méthode de résolution. Plusieurs travaux traitent déjà de ce sujet (Mostaghim *et al.*, 2002; Fieldsend *et al.*, 2003), mais nous pensons que des avancées restent à accomplir dans ce domaine alliant conception et implémentation. À l'heure actuelle, nous travaillons sur l'adaptation de méthodes classiques de la programmation mathématique au sein d'approches métaheuristiques. Ce type de stratégie consiste à réduire, à l'aide d'approches scalaires, le problème d'optimisation multiobjectif à l'étude en un problème monoobjectif. Elles pourraient se montrer utiles à la prise en compte des préférences du décideur, mais également à la recherche de l'ensemble de solutions supportées par le biais de multiples scalarisations, méthodes qui se prêtent naturellement au calcul parallèle. Enfin, nous envisageons de fournir des implémentations de composants dépendants du problème pour des applications classiques de l'optimisation multiobjectif, en particulier des problèmes combinatoires. Ceux-ci seront présentés sous forme de contribution sur le site web de ParadisEO.

Algorithme évolutionnaire SEEA. Lors de nos expérimentations sur le problème de Ring-Star, nous avons mis en avant qu'un algorithme évolutionnaire aussi élémentaire que l'algorithme SEEA, proposé au sein de cette thèse, représentait une alternative intéressante aux algorithmes évolutionnaires classiques de la littérature. Du moins lorsqu'un temps de calcul relativement limité est disponible. À l'avenir, nous envisageons de traiter d'autres problèmes combinatoires à l'aide de cette métaheuristique afin de vérifier si nos observations restent valables, en particulier pour des problèmes où plus de deux fonctions objectif sont impliquées. Pour des temps de calcul plus élevés, nous pensons néanmoins que SEEA pourrait permettre de trouver une première approximation de l'ensemble Pareto optimal, qui servirait ensuite à initialiser la population de départ d'une méthode un peu plus complexe. Pour finir, nous prévoyons d'étudier une version de l'algorithme où la taille de la population créée à chaque itération serait déterminée de façon adaptative, en fonction du nombre d'éléments contenus dans l'archive courante.

Algorithmes de recherche locale DMLS. Des expérimentations supplémentaires sont nécessaires pour analyser l'influence d'autres composants de recherche, intervenants au sein du modèle DMLS, sur le comportement global de la méthode de résolution. Il pourrait en outre se montrer intéressant d'analyser l'influence de ces composants en fonction du paysage du problème étudié, en particulier les caractéristiques liées à la question de connexité entre les solutions Pareto optimales (Ehrgott et Klamroth, 1997).
De façon générale, nous pensons que certaines stratégies du modèle DMLS peuvent constituer des méthodes de résolution attractives concernant l'optimisation multiobjectif combinatoire. En plus de leur relative simplicité de conception et d'implémentation, les algorithmes DMLS requièrent généralement très peu de paramètres. C'est la raison pour laquelle la performance de ces méthodes se doit d'être comparée aux algorithmes évolutionnaires, qui sont souvent considérés comme la référence du domaine, mais qui nécessitent pourtant un plus grand nombre de paramètres. Enfin, des variantes plus évoluées du modèle DMLS, basées sur la recherche tabou ou sur le recuit simulé (Talbi, 2009), pourraient également être étudiées. Il en va de même pour des versions parallèles, distribuées et hybrides. D'après les étude menées par Lust et Teghem (2009) ainsi que Paquete et Stützle (2009) sur le TSP biobjectif, il semble particulièrement avan-

tageux d'hybrider des méthodes de type DMLS avec des méthodes scalaires, pour lesquelles des algorithmes monoobjectif classiques, exacts ou heuristiques, sont directement applicables.

Métaheuristiques coopératives. La combinaison de métaheuristiques fournit généralement des algorithmes de recherche de haute performance pour de nombreuses classes de problèmes académiques et industriels. Bien que ce type d'approche soit de plus en plus présent dans le cadre de l'optimisation multiobjectif, nous pensons que de nombreux schémas de coopérations restent encore à explorer, notamment en hybridant des métaheuristiques à l'aide méthodes exactes (Basseur *et al.*, 2004). Néanmoins, la multiplication de mécanismes évolués a tendance à complexifier la démarche de conception et à augmenter significativement le nombre de paramètres, dont le réglage s'avère souvent délicat. En ce sens, une perspective intéressante serait de définir adaptivement les mécanismes de coopération afin de sélectionner dynamiquement la méthode la plus adaptée selon tel ou tel critère de convergence, ou de diversité. Ce type de coopération est en partie réalisé au sein de l'approche auto-adaptative ACS (*adaptive cooperative search*), mais nécessite d'être généralisée.

Approche parallèle et coopérative basées sur des points de référence multiples. Nous pensons que l'approche parallèle et coopérative basée sur le partitionnement de l'espace objectif à l'aide de points de référence multiples peut susciter de nombreuses orientations pour de futurs travaux. Outre les perspectives déjà mentionnées concernant l'analyse statistique du facteur d'accélération (« *speed-up* ») dû à la parallélisation, il pourrait se montrer intéressant d'étudier le déplacement dynamique des points de référence. En effet, une des questions importantes serait de définir ces points de référence pour qu'il n'y ait pas de chevauchement entre les différentes sous-régions de calcul. Pour cela, il faudrait pouvoir mesurer le chevauchement entre les différents points de référence, et aussi prendre en compte la forme approximée du front Pareto lors de la mise à jour des points de référence. Par ailleurs, chaque spécification de la famille d'*achievement scalarizing functions* prend en paramètre un point de référence unique, et un vecteur de poids associé à chaque objectif. Ce deuxième aspect permettrait de concevoir un algorithme parallèle basé sur un même point de référence et sur différents vecteurs de pondération, et définirait donc un deuxième niveau de parallélisme. Il se montre également utile d'expérimenter cette approche sur des problèmes à plus de deux fonctions objectif. Nous pensons en particulier qu'elle pourrait se montrer performante sur des problèmes où un grand nombre d'objectifs sont considérés (*many-objective optimization*), pour lesquels les métaheuristiques basées sur la dominance Pareto ne semblent pas très performantes en raison du nombre élevé de solutions non-dominées (Wagner *et al.*, 2007). Enfin, une version interactive, où le décideur pourrait simultanément spécifier plusieurs points de référence, et donc plusieurs régions d'intérêts, semble également envisageable.

Optimisation multiobjectif en environnement incertain. Nous l'avons déjà mentionné, le domaine de l'optimisation multiobjectif en environnement incertain n'en est qu'à ses balbutiements. Ainsi, de nombreuses questions restent ouvertes concernant la modélisation, la résolution et l'analyse des performances. En effet, de nombreuses applications requièrent la prise en compte d'incertitude dans le cadre de l'optimisation multiobjectif, autant en ordonnancement (durées d'exécutions, variations des dates dues) que dans les transports (présence incertaine de clients, temps de trajet ou demandes stochastiques) ou encore dans les finances, où la gestion de portefeuille est sujette à un haut risque, et implique des enjeux financiers élevés. Dans ce contexte,

nous avons souligné que les métaheuristiques semblent être un choix pertinent pour lequel, avec un effort relatif, il nous est possible de résoudre des problèmes incertains dont les formulations stochastiques sont très variées.
En outre, lors de nos expérimentations, nous avons pu constater que la prise en compte de l'incertitude s'avérait très coûteuse en temps de calcul, du fait de l'évaluation multiple des solutions et de l'augmentation de la cardinalité des ensembles à comparer. Aussi, nous pensons que le calcul parallèle a indéniablement un rôle à jouer pour la résolution de tels problèmes, en particulier pour les applications réelles où l'étape d'évaluation requiert déjà en elle-même d'importantes ressources calculatoires. Par exemple, un modèle de parallélisation immédiat, qui pourrait être réalisé à court terme, serait de distribuer le calcul des différentes évaluations à exécuter pour chacune des solutions.

Métaheuristiques pour l'optimisation multiobjectif interactive. Au cours des vingt dernières années, de nombreuses avancées fondamentales, algorithmiques et applicatives ont été réalisées afin de trouver une bonne approximation de l'ensemble Pareto optimal à l'aide de métaheuristiques, et en particulier d'algorithmes évolutionnaires. La communauté EMO (*evolutionary multiobjective optimization*) est toujours très active, et en perpétuelle croissance. Or, ce domaine de recherche a occulté tout un pan du processus de l'optimisation multiobjectif. La résolution d'un problème d'optimisation multiobjectif consiste en effet à trouver la solution Pareto optimale qui répond le mieux aux préférences du décideur, ce qui s'avère donc principalement être une question de prise de décision. Il est vrai qu'avoir une bonne approximation de l'ensemble Pareto optimal à disposition fournit une information capitale aux yeux du décideur. Ceci étant, cette étape doit reprendre place au sein d'un processus plus général de prise de décision, et d'articulation des préférences. Ainsi, la communauté EMO renouera avec le domaine de la prise de décision multicritère (MCDM), champs de recherche bien établi et plein d'expériences. L'incorporation des préférences du décideur dans le processus de résolution résume l'une des activités principales de la communauté MCDM. Certaines méthodes visent à atteindre une solution qui répond aux préférences du décideur, alors que d'autres visent à progressivement obtenir des informations préférentielles auprès du décideur afin de converger vers des solutions privilégiées. Pourtant, jusqu'ici, très peu d'études ont fait état de l'incorporation des préférences du décideur au sein de métaheuristiques pour l'optimisation multiobjectif.
Récemment, plusieurs approches combinant les idées provenant des deux communautés EMO et MCDM ont fait leur apparition ; voir par exemple le livre édité par Branke *et al.* (2008). L'algorithme que nous avons utilisé au sein de l'approche à points de référence multiples en est une bonne illustration (Thiele *et al.*, 2009). Une telle métaheuristique vise à obtenir un ensemble de solutions alternatives dont la projection dans l'espace objectif se situe aux alentours d'un point de référence donné par le décideur ; d'autres méthodes similaires ont également été proposées dans le même contexte (Molina *et al.*, 2009). Aussi, au sein de métaheuristiques pour l'optimisation multiobjectif interactive, le décideur spécifie ses préférences progressivement durant le processus de recherche. La question cruciale concerne alors le type d'information échangée entre le décideur et la métaheuristique. Ces méthodes hybrides entre EMO et MCDM possèdent de nombreux avantages en relation avec l'efficacité des méthodes en termes de temps de calcul, et avec le respect des choix du décideur. Nous pensons que l'augmentation de ces études va permettre d'accroître la coopération entre les chercheurs en informatique et en MCDM, et de tirer parti des avancées dans ces deux domaines au cours des années à venir.

Annexe

Implémentation de métaheuristiques pour l'optimisation multiobjectif sous ParadisEO-MOEO

Dans cette annexe, une description détaillée des classes de base fournies dans la plateforme ParadisEO [4] pour résoudre un problème d'optimisation multiobjectif est donnée.
Puisque l'implémentation d'une métaheuristique pour l'optimisation multiobjectif ne diffère de celle d'une métaheuristique pour l'optimisation mono-objectif que pour un certain nombre de points, certains composants de ParadisEO-EO et de ParadisEO-MO sont directement réutilisables. Par conséquent, notez que, dans le texte qui suit, le nom de toutes les classes de ParadisEO-EO est préfixé par `eo`, celui des classes de ParadisEO-MO est préfixé par `mo`, alors que les classes de ParadisEO-MOEO est préfixé par `moeo`. ParadisEO est une plateforme orientée-objet, ses composants seront donc spécifiés selon le standard UML [5]. Ainsi, les diagrammes UML principaux sont présentés. Le diagramme d'héritage complet ainsi que la documentation des classes et de nombreux exemples d'utilisation sont disponibles sur le site de ParadisEO. En outre, une grande partie des composants de ParadisEO sont fondés sur la notion de Template. Ce concept et de nombreuses fonctionnalités connexes sont proposés par le langage de programmation C++ et permet de manipuler des classes génériques, de telle sorte qu'elles peuvent fonctionner avec des types de donnée différents sans avoir à être ré-implémentées pour chacun d'entre eux.

Par la suite, nous allons présenter à la fois les composants dépendants et indépendants du problème à résoudre. Ainsi, les composants spécifiques au problème (représentation, évaluation, initialisation, variation, et voisinage) et les composants génériques communs (relation de dominance, affectation des valeurs de fitness, affectation des valeurs de diversité, gestion de l'archive, condition d'arrêt et outils statistiques) sont suivis par les composants génériques liés aux algorithmes évolutionnaires (sélection, remplacement) et les composants génériques liés aux algorithmes de recherche locale (sélection de l'ensemble courant, exploration du voisinage).

Composants spécifiques au problème

Ici sont présentés les composants spécifiques à la résolution d'un problème particulier. Ainsi, une présentation des composants inévitables de représentation, d'évaluation et d'initialisation sont suivis par une présentation des composants de variation à implémenter pour les algorithmes évolutionnaires, et par une présentation des composants de voisinage à implémenter pour les algorithmes de recherche locale.

Représentation. Une solution doit à la fois être représentée dans les espaces décisionnel et objectif. Bien que la représentation dans l'espace objectif puisse être considérée comme indépendante du problème, la représentation dans l'espace de décision doit être pertinente vis-à-vis du problème abordé.

4. `http://paradiseo.gforge.inria.fr`.
5. OMG Unified Modeling Language Specification.

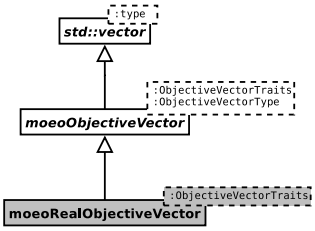


Fig. 4.11 – Diagramme UML pour la représentation d'une solution dans l'espace objectif.

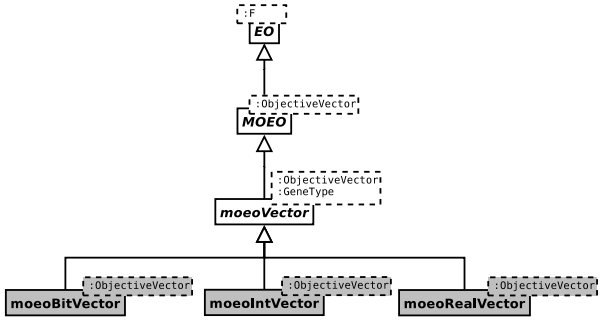


Fig. 4.12 – Diagramme UML pour la représentation complète d'une solution.

Sous ParadisEO-MOEO, l'implémentation d'une représentation est probablement l'étape la plus coûteuse parmi tous les composants nécessaires à la résolution du problème souhaité. Celle-ci se décompose en trois étapes. Tout d'abord, il est nécessaire de définir le nombre de fonctions objectif souhaité et, pour chacune d'elle, si elle est à minimiser ou à maximiser. Ceci peut être réalisé à l'aide la classe statique `moeoObjectiveVectorTraits`.
Dans un second temps, une classe héritant de `moeoObjectiveVector` doit être créée pour la représentation d'un vecteur objectif, comme illustré sur la figure 4.11. Étant donné qu'une majorité des problèmes d'optimisation multiobjectif traitent de valeurs réelles, une classe de modélisation de vecteurs objectif réels est également proposée (`moeoRealObjectiveVector`), et peut donc être utilisée pour tout problème sans perte de généralité.
Enfin, la classe à implémenter pour la représentation globale d'une solution dans ParadisEO-MOEO doit étendre la classe MOEO. Cette modélisation tend à être applicable pour n'importe quel type de problème et a pour but d'être la plus générale possible. Néanmoins, ParadisEO-MOEO prévoit également des classes directement instantiables pour un nombre de représentations standard. En particulier, des implémentations basées sur des vecteurs de bits, d'entiers ou de réels sont proposées, et peuvent donc être directement utilisées au sein d'une application conçue sous ParadisEO-MOEO. Ces classes sont résumées sur la figure 4.12.
Par ailleurs, notez que la gestion d'une population est réalisée à l'aide d'un objet de type `eoPop`, et peut donc être vu comme un vecteur dont les éléments sont de type `MOEO`.

Évaluation. La façon d'évaluer une solution dans l'espace objectif doit être assurée par un élément héritant de la classe abstraite `eoEvalFunc`, dont une illustration est donnée sur la figure 4.13. Il a pour but essentiel de fixer les valeurs objectif d'un objet de type `MOEO`. Notez que des implémentations fournies permettent par exemple de transformer une fonction spécifique utilisant des pointeurs d'objet (`eoEvalFuncPtr`), ou encore une fonction implémentée à l'extérieure de ParadisEO telle qu'une fonction « boîte-noire » (`eoExternalEvalFunc`). Enfin, la classe `eoEvalFuncCounter` permet de sauvegarder le nombre d'évaluations effectuées par une fonctioèn d'évaluation passée en paramètre, ceci afin de pouvoir définir le critère d'arrêt comme un nombre maximum d'évaluations ou de simples fins statistiques.
En règle générale, pour des problèmes d'optimisation réels, l'évaluation d'une solution dans l'espace objectif est de loin l'étape la plus coûteuse en temps de calcul de l'approche métaheuristique choisie. Une façon de surmonter ce problème est l'utilisation de modèles parallèles et distribués.

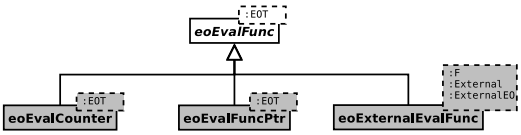


Fig. 4.13 – Diagramme UML pour l'évaluation

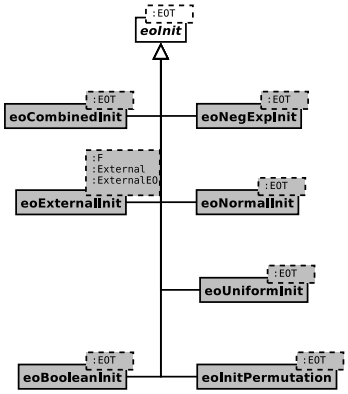


Fig. 4.14 – Diagramme UML pour l'initialisation

L'implémentation d'une évaluation parallèle est largement simplifiée dans le cadre de ParadisEO grâce au module ParadisEO-PEO, où l'évaluation parallèle de la population peut par exemple se définir de façon transparente par simple instanciation.

Initialisation. Afin d'initialiser une solution ou une population de solutions, une stratégie d'initialisation, dépendante de la représentation choisie doit hériter de la classe `eoInit`. Plusieurs techniques d'initialisation existent déjà dans beaucoup de bibliothèques standards pour des représentations classiques, ce qui est également le cas avec ParadisEO. Mais certaines situations exigent une implémentation spécifique ou une combinaison de plusieurs opérateurs. Tel qu'indiqué sur la figure 4.14, la plateforme offre une gamme de stratégies d'initialisation héritant toutes de `eoInit`, ainsi que d'une façon commode de les combiner à l'aide d'un objet de type `eoCombinedInit`.

Variation. Les opérateurs de variation doivent tous dériver de la classe de base `eoOp` (Fig. 4.15). Quatre classes abstraites héritent de `eoOp`, à savoir `eoMonOp` pour les opérateurs de mutation, `eoBinOp` et `eoQuadOp` pour les opérateurs de croisement binaires et quadratiques, et `eoGenOp` pour les autres types d'opérateur plus généraux. Un ensemble d'opérateurs de même arité peuvent également être facilement combinés proportionnellement à des poids respectifs. Certains mécanismes classiques de variation (pour des représentations basées sur des bits, des entiers, des réels ou des permutations) sont également proposés au sein de la plateforme. En outre, un mécanisme d'hybridation peut aisément être conçu en utilisant une métaheuristique à base de solution unique en lieu et place de l'opérateur de mutation, voir Paradiseo-MO (Boisson *et al.*, 2009). L'ensemble de tous les opérateurs de variation pour un problème donné doit être intégré dans un objet de type `eoTranform` afin d'être utilisé par l'algorithme évolutionnaire.

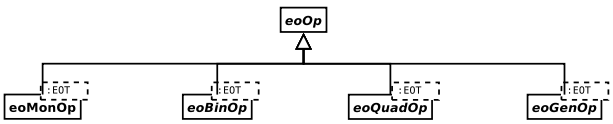


Fig. 4.15 – Diagramme UML pour les opérateurs de variation

Voisinage. La structure du voisinage est basée sur la notion de mouvements. Dans un premier temps, il s'avère donc nécessaire de définir un mouvement applicable à la représentation choisie pour résoudre le problème traité. Celui-ci doit hériter de la classe abstraite `moMove`. Puis, une classe de type `moMoveInit` doit être définie afin de pouvoir initialiser le voisinage d'une solution donnée. Un tel objet doit permettre de définir le mouvement initial d'une solution donnée. Ensuite, un objet de type `moNextMove` permet de se déplacer au sein du voisinage en cours d'exploration. Il se doit de déterminer s'il reste des voisins à explorer et, le cas échéant, d'accéder au voisin suivant. Enfin, un objet de type `moIncrEval` permet d'évaluer le voisin en cours d'exploration de façon incrémentale, ceci à l'aide de la solution courante, et du mouvement à appliquer. Une structure de voisinage pour un problème donné est donc définie, en termes d'implémentations, pour les 4 classes : `moMove`, `moMoveInit`, `moNextMove` et `moIncrEval`. Notez qu'à l'exception de l'évaluation incrémentale, où plusieurs valeurs objectif sont dorénavant à considérer, ces composants n'ont rien de spécifique à l'optimisation multiobjectif et proviennent donc du module ParadisEO-MO (Boisson *et al.*, 2009).

Composants génériques communs

Les composants indépendants du problème partagés par les algorithmes évolutionnaires et les algorithmes de recherche locale multiobjectif sont ceux de la relation de dominance, de l'affectation d'une valeur de fitness, de la préservation de la diversité, de la gestion de l'archive, et enfin de la condition d'arrêt et d'éventuels outils statistiques.

Relation de dominance. Le choix d'une relation de dominance ne prend pas directement part au sein de l'algorithme implémenté, mais peut être vu comme un composant de bas niveau utilisé par d'autres composants génériques de plus haut niveau tels que les stratégies d'affectation d'une valeur de fitness, ou de gestion de l'archive. La relation de dominance Pareto est la plus courante, et est donc généralement employé par défaut dans chacune de ces stratégies. Mais d'autres relations peuvent être considérées. Et bien sûr, des relations de dominance différentes peuvent être définies à des étapes distinctes de l'algorithme, afin par exemple de définir plusieurs niveaux de pression. Une relation de dominance permet de comparer deux vecteurs objectif afin de déceler si l'un domine l'autre. Comme illustré dans le diagramme UML de la figure 4.16, les relations existantes au sein de ParadisEO-MOEO sont celles de dominance faible, de dominance stricte et d'ϵ-dominance telles que définies dans la section 1.1.3, ainsi que la relation de g-dominance proposée par Molina *et al.* (2009) et basée sur un point de référence.

Affectation d'une valeur de fitness. Les stratégies d'affectation de valeurs de fitness les plus couramment rencontrées sont implémentées dans Paradiseo-MOEO : les approches scalaires, les approches fondées sur une relation de dominance et les approches basées sur un indicateur de qualité. Suivant la classification présentée dans la section 2.1.3.1, elles sont ici classées en quatre catégories, comme illustré dans le diagramme UML de la figure 4.17 :
- approches scalaires : `moeoScalarFitnessAssignment` ;
- approches basées sur un critère : `moeoCriterionBasedFitnessAssignment` ;
- approches basées sur une relation de dominance : `moeoDominanceBasedFitnessAssignment` ;
- approches basées sur un indicateur : `moeoIndicatorBasedFitnessAssignment`.

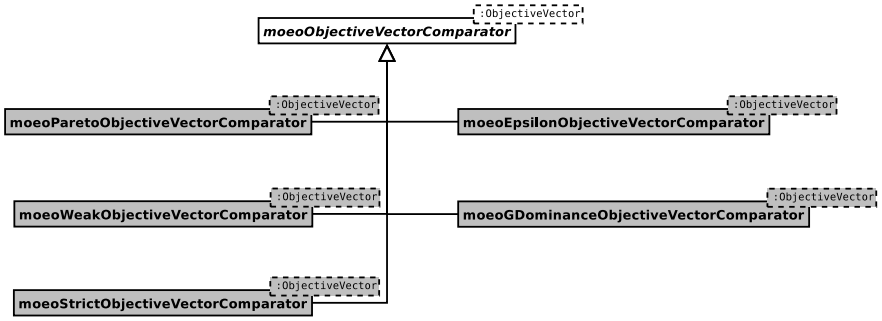


Fig. 4.16 – Diagramme UML pour les relations de dominance (comparaison deux-à-deux de vecteurs objectif).

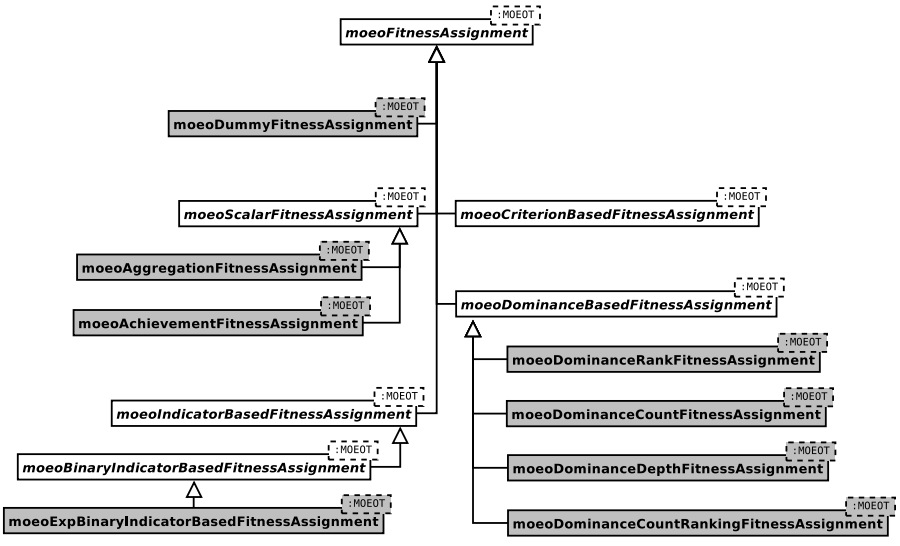


Fig. 4.17 – Diagramme UML pour l'affectation d'une valeur de fitness.

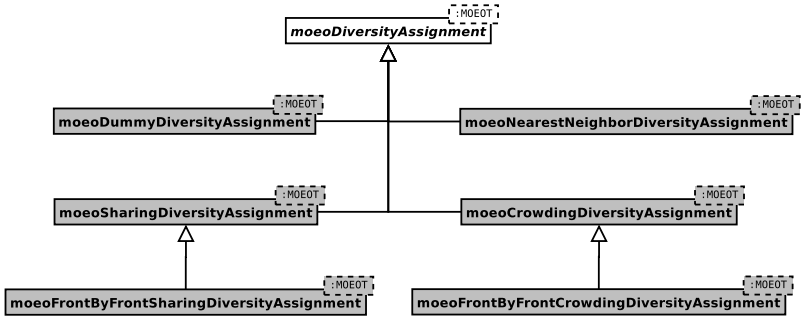


Fig. 4.18 – Diagramme UML pour l'affectation d'une valeur de diversité.

Les stratégies concrètes implémentées sont détaillées ci-dessous. En outre, une classe fictive (`moeoDummyFitnessAssignment`) est également proposée dans le cas d'une éventuelle utilité pour certaines implémentations.

Approches scalaires. Deux approches scalaires sont proposées. Tout d'abord, une simple stratégie d'agrégation des fonctions objectif, basée sur un vecteur de poids, est implémentée dans la classe `moeoAggregationFitnessAssignment`. La deuxième technique scalaire est la famille des *achievement scalarizing functions* (`moeoAchievementFitnessAssignment`) proposée par Wierzbicki (1980). Cette approche scalaire, basée sur un point de référence arbitraire généralement donnée par le décideur, est utilisé dans de nombreuses méthodes MCDM.

Approches basées sur une relation de dominance. Les trois approches basées sur une relation de dominance présentées lors de la section précédente, à savoir le rang de dominance, le compte de dominance, et la profondeur de dominance, sont proposées dans la plateforme. De plus, une stratégie hybride entre le rang et le compte de dominance, où la valeur de fitness d'une solution x correspond à la somme des rangs de toutes les solutions dominées par x, est également implémentée. Par exemple, elle est utilisée dans SPEA2 (Zitzler *et al.*, 2001b). Notez que la relation de dominance utilisée par défaut par toutes ces stratégies est la dominance-Pareto, mais n'importe quelle autre relation de dominance peut être utilisée par simple passage de paramètre.

Approches basées sur un indicateur. Pour finir, une approche basée sur un indicateur de qualité est implémentée. Il s'agit de la méthode utilisée au sein de l'algorithme IBEA (Zitzler et Künzli, 2004). L'indicateur binaire de qualité pris en compte lors du calcul des valeurs de fitness est passé en paramètre, et plusieurs indicateurs sont proposés dans Paradiseo-MOEO.

Préservation de la diversité. La figure 4.18 présente les stratégies de préservation de la diversité implémentées. Celles-ci héritent toutes de la classe `moeoDiversityAssignment`. Ainsi, en plus d'une technique fictive, les méthodes de *sharing*, de *crowding*, et du $k$ème plus proche voisin sont proposées. Notez que des alternatives aux procédures de *sharing* et de *crowding* traditionnelles opèrent sur des sous-ensembles de la population ayant une même valeur de fitness. De telles stratégies sont utilisées par certains algorithmes évolutionnaires (Fonseca et Fleming, 1993; Srinivas et Deb, 1994; Deb *et al.*, 2002).

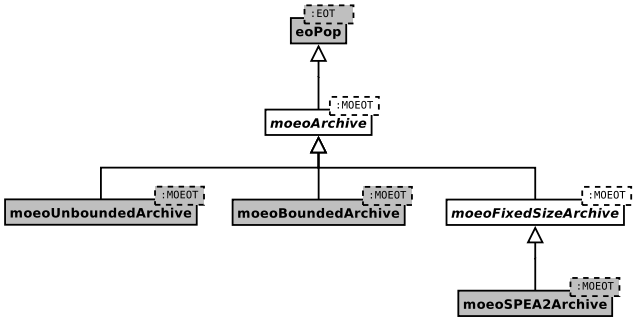


Fig. 4.19 – Diagramme UML pour la gestion de l'archive.

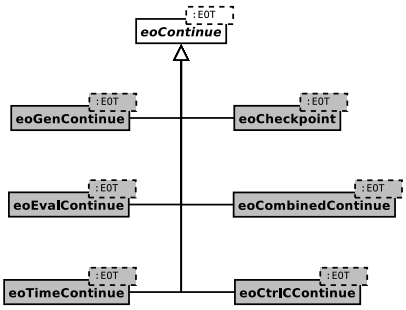


Fig. 4.20 – Diagramme UML pour la condition d'arrêt.

Gestion de l'archive. Une archive, représentée par la classe abstraite `moeoArchive`, est une population utilisant une relation de dominance pour mettre son propre contenu à jour. Une classe abstraite pour les archives de taille fixe est donnée : `moeoFixedSizeArchive`. Mais des implémentations d'une archive de taille non bornée (`moeoUnboundedArchive`), d'une archive générique de taille bornée basée sur une stratégie de fitness et de diversité (`moeoBoundedArchive`), ainsi que la stratégie d'archivage utilisée au sein de l'algorithme SPEA2 (`moeoSPEA2Archive`) sont fournies. Le diagramme d'héritage des techniques d'archivage est donné sur la figure 4.19.

En règle générale, la relation de dominance Pareto est utilisée pour mettre l'archive à jour, et est donc employée par défaut. Mais, d'autres critères peuvent être utilisés par simple passage de paramètre. Par conséquent, la plateforme offre la possibilité d'utiliser toute autre relation de dominance par le biais d'un objet de type `moeoObjectiveVectorComparator` (Fig. 4.16).

Condition d'arrêt. Dans le cadre de ParadisEO, de nombreux critères d'arrêt, héritant de la classe abstraite `eoContinue`, sont fournis (Fig. 4.20). Par exemple, l'algorithme peut s'arrêter après un certain nombre d'itérations (`eoGenContinue`), un certain nombre d'évaluations (`eoEvalContinue`), un temps d'exécution (`eoTimeContinue`), ou encore de façon interactive, dès que l'utilisateur le décide (`eoCtrlCContinue`). En outre, différents critères d'arrêt peuvent être facilement combinés à l'aide d'un objet de type `eoCombinedContinue`. Dans ce cas, le processus de recherche s'arrête dès que l'un des critères est satisfait.

Outils statistiques. À chaque itération de l'algorithme principal, de nombreuses autres procédures peuvent aisément être appelées. En effet, la classe `eoCheckPoint` permet d'effectuer certaines actions systématiques à chaque itération de l'algorithme de façon transparente, en étant intégrée dans le critère d'arrêt utilisé. Ce moteur est particulièrement utile pour les mécanismes de tolérance aux pannes, mais aussi dans le but de calculer diverses valeurs statistiques.
En effet, de nombreux outils statistiques sont également fournis dans ParadisEO-MOEO. Il est par exemple possible d'enregistrer le contenu de l'archive à chaque itération, de sorte que l'évolution de l'approximation de l'ensemble non-dominé peut être observée ou étudiée à l'aide d'outils graphiques comme GUIMOO [6]. En outre, une question importante dans le domaine de l'optimisation multiobjectif concerne l'analyse de performance et les indicateurs de qualité.

6. GUIMOO est une interface graphique pour l'optimisation multiobjectif disponible à l'URL : `http://guimoo.gforge.inria.fr/`.

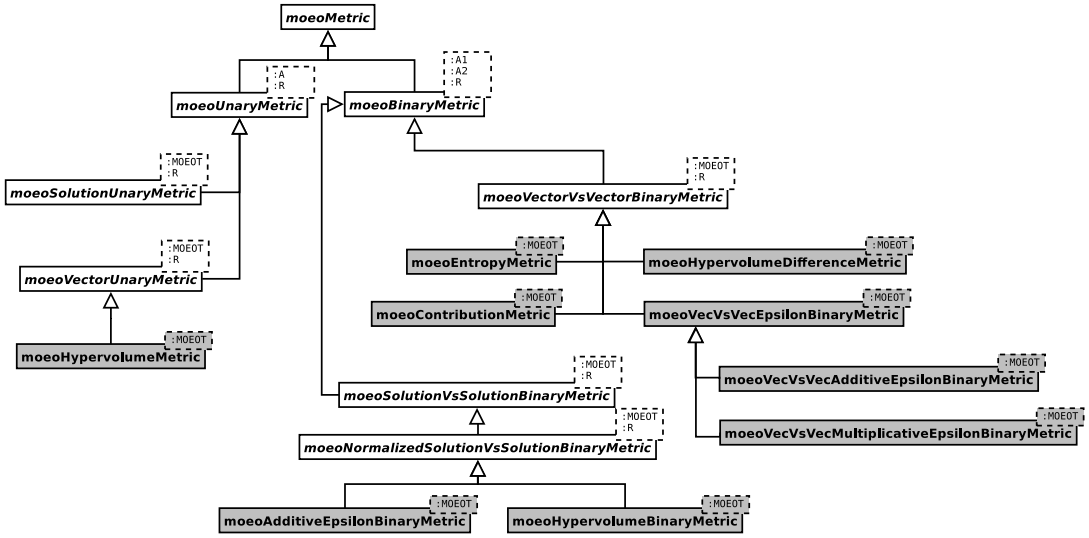


Fig. 4.21 – Diagramme UML pour les indicateurs de qualité.

Plusieurs indicateurs sont présents (Fig. 4.21), comprenant la métrique hypervolume à la fois sous sa forme unaire et binaire, la métrique entropie, la métrique contribution ainsi que les indicateurs epsilon additif et multiplicatif. Une autre caractéristique intéressante est la possibilité de comparer l'archive courante à l'archive de l'itération précédente au moyen d'une métrique binaire, et d'imprimer la progression de cette mesure itération après itération.

Composants génériques pour les algorithmes évolutionnaires

Les composants génériques communs aux algorithmes évolutionnaires sont ceux de sélection et de remplacement.

Sélection. Quatre stratégies de sélection sont fournies dans Paradiseo-MOEO. Tout d'abord, la stratégie de sélection aléatoire (`moeoRandomSelectOne`) consiste à choisir une solution Parent au hasard parmi les membres de la population, sans prendre aucune information de fitness de diversité en compte. Ensuite, le tournoi de sélection déterministe (`moeoDetTournamentSelectOne`) consiste à réaliser un tournoi entre m solutions choisies aléatoirement au sein de la population, et à choisir le meilleur de ces individus comme Parent. Troisièmement, le tournoi de sélection stochastique (`moeoStochTournamentSelectOne`) consiste à réaliser un tournoi binaire entre deux individus choisis au hasard parmi les membres de la population, et à choisir le meilleur des deux avec une probabilité p, ou le moins bon des deux, avec une probabilité $(1-p)$. Enfin, la sélection élitiste (`moeoSelectOneFromPopAndArch`) consiste à sélectionner un individu de la population avec une probabilité p ou un individu de l'archive avec une population $(1-p)$. Chacune des sous-stratégies de sélection parmi la population et parmi l'archive peut être définie par une des trois autres stratégies présentées ci-dessus. Ainsi, les solutions élites participent activement à la recherche en contribuant au moteur d'évolution. Toutes ces stratégies de sélection sont du type `moeoSelectOne` et permettent de sélection un seul individu Parent. Une telle méthode doit donc être intégrée dans un objet de type `eoSelect` pour pouvoir sélectionner le nombre souhaité de

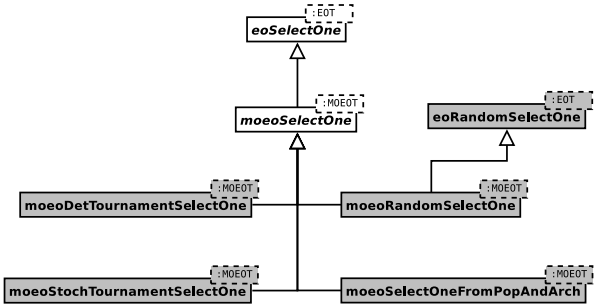


Fig. 4.22 – Diagramme UML pour la sélection

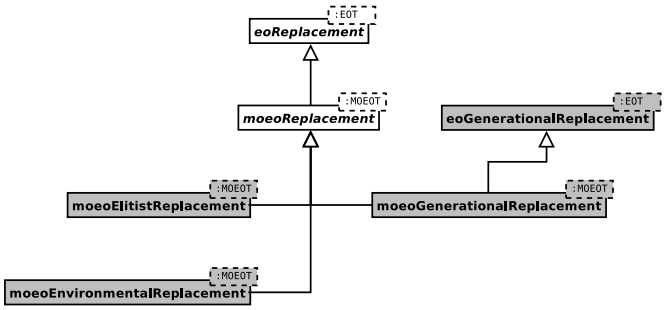


Fig. 4.23 – Diagramme UML pour le remplacement

solutions Parent par rapport à une valeur fixe ou encore par rapport à un pourcentage de la taille de la population courante.

Remplacement. À l'heure actuelle, trois stratégies de remplacement sont proposées. Tout d'abord, le remplacement générationnel (`moeoGenerationalReplacement`) consiste à ne conserver que les solutions issues de la population Enfant, les individus Parent sont donc tous supprimés. Ensuite, le remplacement élitiste préserve les meilleures solutions provenant des deux populations. Une première stratégie élitiste (`moeoElitistReplacement`) consiste à réaliser une réduction directe en supprimant tous les moins bons individus d'un coup. Une deuxième stratégie élitiste (`moeoEnvironmentalReplacement`) consiste à itérativement supprimer la moins bonne solution, et à mettre à jour les informations de fitness et de diversité des individus restants ; ceci jusqu'à ce que la taille attendue de la population soit atteinte.

Algorithmes évolutionnaires multiobjectif

Maintenant que tous les composants liés au problème, à l'optimisation multiobjectif et à l'optimisation évolutionnaire sont définis, un algorithme évolutionnaire multiobjectif peut facilement être conçu et implémenté en utilisant les classes de ParadisEO. Étant donné que l'implémentation est conceptuellement divisée en composants, différents opérateurs peuvent être facilement expérimentés sans engendrer d'importantes modifications en termes d'écriture de code. Comme nous l'avons vu auparavant, une vaste gamme de composants est déjà proposée. Mais, il faut garder à l'esprit que cette liste n'est pas exhaustive, car la plateforme est en perpétuelle évolution et offre tout ce qui est nécessaire pour développer de nouveaux composants avec un minimum d'effort.

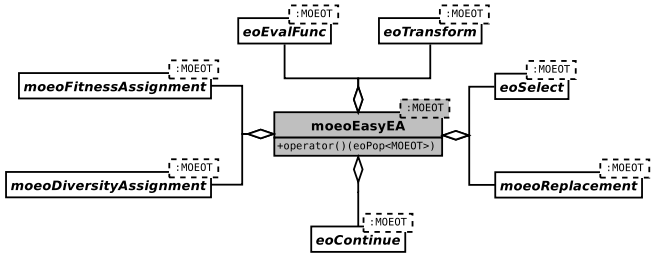


Fig. 4.24 – Une instantiation possible d'un algorithme évolutionnaire multiobjectif.

En effet, ParadisEO est une plateforme « boîte blanche » qui tend à être la plus flexible possible tout en offrant l'environnement le plus simple et convivial possible pour un utilisateur novice.

Le fondement de la classe `moeoEasyEA` est illustré sur la figure 4.24. Celle-ci permet de définir un algorithme évolutionnaire multiobjectif de façon générique, en respectant le modèle d'unification présenté à la section précédente. Il suffit de préciser tous les composants requis à son instanciation. Toutes les classes utilisent un Template en paramètre (`MOEOT`) qui symbolise la représentation d'une solution pour le problème traité. Cette représentation doit être implémentée comme héritant de la classe MOEO tel que nous l'avons décrit dans la section 4.4.2.4. Le composant relatif à la gestion de l'archive ne figure pas dans le diagramme UML. Nous avons en effet délibérément choisi de permettre l'utilisation d'une archive comme optionnelle. Le gestionnaire de mise à jour de l'archive peut facilement être intégré au sein de l'algorithme par le biais du processus de Checkpointing. De même, l'étape d'initialisation n'apparaît pas non plus, car une exécution de la méthode principale d'une instance de la classe `moeoEasyEA` débute avec une population déjà initialisée.

Toujours en gardant à l'esprit de satisfaire à la fois l'utilisateur novice et l'utilisateur plus expérimenté, ParadisEO-MOEO prévoit également des classes d'algorithmes évolutionnaires multiobjectif encore plus faciles à utiliser que `moeoEasyEA`, voir la figure 2.1.4.3. Ces classes proposent des implémentations d'algorithmes classiques en utilisant les composants de base de ParadisEO. Ce qui valide la grande flexibilité de la plateforme. Ils sont basés sur une combinaison simple des composants. Ainsi, les algorithmes MOGA (Fonseca et Fleming, 1993), NSGA (Srinivas et Deb, 1994), NSGA-II (Deb *et al.*, 2002), SPEA2 (Zitzler *et al.*, 2001b), IBEA (Zitzler et Künzli, 2004) et SEEA (Liefooghe *et al.*, 2008e) sont proposés de telle sorte qu'un nombre minimum de paramètres spécifiques au problème ou à l'algorithme soient nécessaires. Par exemple, pour instancier NSGA-II pour un problème multiobjectif continu original, il est possible d'utiliser les opérateurs de représentation, d'initialisation et de variation fournis, l'évaluation étant donc le seul composant à implémenter. Il en va de même pour n'importe quel autre de ces algorithmes et pour n'importe quel problème pouvant être représenté à l'aide d'un vecteur de bits, d'entiers ou encore d'une permutation. Ces algorithmes faciles à utiliser tendent à être utilisés comme références dans le monde académique, ceci afin de pouvoir équitablement comparer les performances de différentes méthodes, leur implémentation partageant les mêmes composants de base. Néanmoins, elles sont également bien adaptées à une application directe de résolution d'un problème du monde réel. Dans un proche avenir, d'autres métaheuristiques évolutionnaires multiobjectif faciles à utiliser seront proposées, au fur et à mesure que de nouveaux composants seront implémentés dans ParadisEO-MOEO.

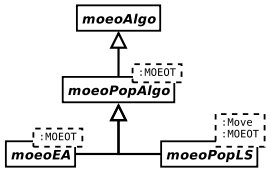


Fig. 4.25 – Diagramme UML pour les algorithmes multiobjectif.

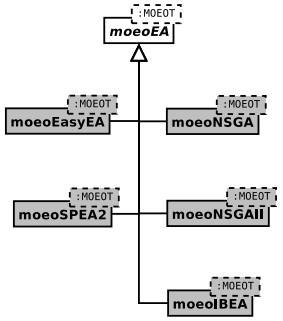


Fig. 4.26 – Diagramme UML pour les algorithmes évolutionnaires multiobjectif.

Composants génériques pour les algorithmes de recherche locale

Les composants indépendants du problème prenant part à la conception et à l'implémentation d'une recherche locale de type DMLS sont liés à la sélection de l'ensemble courant et à l'exploration du voisinage.

Sélection de l'ensemble courant. Deux stratégies simples de sélection de l'ensemble courant parmi les solutions non-visitées d'une archive sont fournies dans ParadisEO-MOEO (Fig. 4.27). Premièrement, la stratégie exhaustive (`moeoExhaustiveUnvisitedSelect`) consiste à sélectionner l'ensemble des solutions non-visitées de l'archive. Deuxièmement, une stratégie alternative partielle (`moeoNumberUnvisitedSelect`) consiste à sélectionner un sous-ensemble des solutions non-visitées de l'archive aléatoirement, à hauteur d'une valeur passée en paramètre (par défaut, 1).

Exploration du voisinage. Il existe actuellement, au sein de ParadisEO-MOEO, quatre techniques directement insatiables pour l'exploration du voisinage d'une population de solutions fournie en entrée. Elles héritent toutes de la classe `moeoPopNeighborhoodExplorer` et sont subdivisées en deux classes (Fig. 4.28). Tout d'abord, la stratégie d'exploration exhaustive (`moeoExhaustiveNeighborhoodExplorer`) consiste à évaluer la totalité du voisinage des solutions. Ensuite, trois stratégies d'exploration partielle, héritant de `moeoSubNeighborhoodExplorer`, sont proposées : une exploration consistant à n'explorer que les N premiers voisins de chaque solution (`moeoSimpleSubNeighborhoodExplorer`), une autre consistant, pour chaque solution courante x, à explorer le voisinage jusqu'à ce qu'une solution non-dominée par rapport à x ne soit trouvée (`moeoNoDesimprovingNeighborhoodExplorer`), et une dernière semblable à la précédente mais dont le processus d'exploration est arrêté lorsqu'une solution dominant x est trouvée (`moeoFirstImprovingSubNeighborhoodExplorer`). Bien sûr, une relation de dominance arbitraire peut être passée en paramètre, et la relation de dominance Pareto est utilisée par défaut.

Algorithmes de recherche locale multiobjectif

La totalité des composants requis pour la conception et l'implémentation d'un algorithme de type DMLS a été présentée auparavant. Grâce à cette décomposition fine, il est maintenant possible de

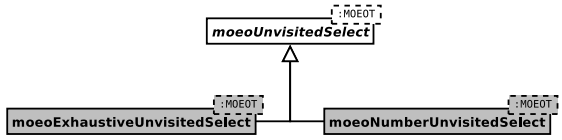


Fig. 4.27 – Diagramme UML pour la sélection de l'ensemble courant.

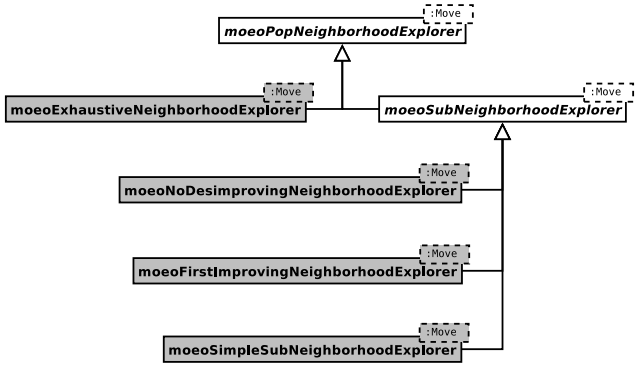


Fig. 4.28 – Diagramme UML pour l'exploration du voisinage.

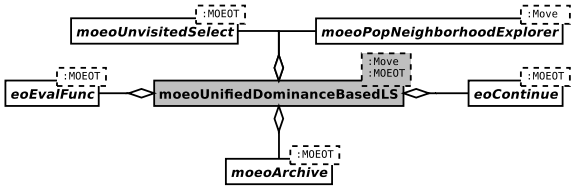


Fig. 4.29 – Une instantiation possible d'un algorithme de recherche locale de type DMLS.

développer un tel algorithme par simple instanciation, comme illustré sur la figure 4.29. Notez que différentes stratégies peuvent également être testées sans effort en modifiant les composants indépendants du problème qui sont utilisés. Ainsi, la classe `moeoUnifiedDominanceBasedLS` requiert la définition d'un Template symbolisant la représentation choisie pour le problème à résoudre (`MOEOT`) et d'un Template représentant le voisinage implémenté (`Move`).
Par ailleurs, notez que des implémentations directes d'algorithmes classiques, tels que les algorithmes PLS-1 et PLS-2 présentés lors de la section 2.1.5.3, ainsi que l'algorithme IBMOLS sont également proposées (Fig. 4.30). Ces classes utilisent les composants de base de ParadisEO. Et nous espérons que de nombreux algorithmes de recherche locale multiobjectif additionnels vont rapidement être proposés au sein de la plateforme.

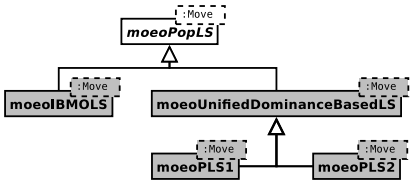


Fig. 4.30 – Diagramme UML pour les algorithmes de recherche locale multiobjectif.

Bibliographie

Aguirre, H. et Tanaka, K. (2005). Random bit climbers on multiobjective MNK-landscapes : Effects of memory and population climbing. *IEICE Trans. Fundamentals*, 88-A(1):334–345.

Akinc, U. et Srikanth, K. (1992). Optimal routing and process scheduling for a mobile service facility. *Networks*, 22(2):163–183.

Angel, E., Bampis, E. et Gourvés., L. (2004). A dynasearch neighbohood for the bicriteria traveling salesman problem. *In Metaheuristics for Multiobjective Optimisation*, volume 535 de *Lecture Notes in Economics and Mathematical Systems*, chapitre 6, pages 153–176. Springer-Verlag, Berlin, Germany.

Babbar, M., Lakshmikantha, A. et Goldberg, D. E. (2003). A modified NSGA-II to solve noisy multiobjective problems. *In Genetic and Evolutionary Computation Conference (GECCO 2003)*, volume 2723 de *Lecture Notes in Computer Science*, pages 21–27, Chicago, Illinois, USA. Springer-Verlag.

Bader, J. et Zitzler, E. (2008). HypE : An algorithm for fast hypervolume-based many-objective optimization. TIK Report 286, Computer Engineering and Networks Laboratory (TIK), ETH Zurich.

Baldacci, R., Dell'Amico, M. et Salazar Gonzàlez, J. J. (2007). The capacitated m-ring star problem. *Operations Research*, 55(6):1147–1162.

Barrico, C. et Antunes, C. H. (2007). An evolutionary approach for assessing the degree of robustness of solutions to multi-objective models. *In* Yang, S., Ong, Y.-S. et Jin, Y., éditeurs : *Evolutionary Computation in Dynamic and Uncertain Environments*, volume 51 de *Studies in Computational Intelligence*, chapitre 25, pages 565–582. Springer-Verlag.

Basseur, M. (2005). *Conception d'algorithmes coopératifs pour l'optimisation multi-objectif : Application aux problèmes d'ordonnancement de type Flow-shop*. Thèse de doctorat, Université des Sciences et Technologies de Lille, France.

Basseur, M. et Burke, E. K. (2007). Indicator-based multi-objective local search. *In IEEE Congress on Evolutionary Computation (CEC 2007)*, pages 3100–3107, Singapore. IEEE Press.

Basseur, M., Lemesre, J., Dhaenens, C. et Talbi, E.-G. (2004). Cooperation between branch and bound and evolutionary approaches to solve a biobjective flow shop problem. *In Third International Worshop on Evolutionary Algorithms (WEA 2004)*, volume 3059 de *Lecture Notes in Computer Science*, pages 72–86, Angra dos Reis, Brazil. Springer-Verlag.

Basseur, M., Seynhaeve, F. et Talbi, E. (2003). Adaptive mechanisms for multiobjective evolutionary algorithms. *In Congress on Engineering in System Application (CESA 2003)*, pages 72–86, Lille, France.

BASSEUR, M., SEYNHAEVE, F. et TALBI, E.-G. (2002). Design of multi-objective evolutionary algorithms : Application to the flow shop scheduling problem. *In IEEE Congress on Evolutionary Computation (CEC 2002)*, pages 1151–1156, Piscataway, NJ, USA. IEEE Press.

BASSEUR, M., TALBI, E.-G., NEBRO, A. et ALBA, E. (2006). Metaheuristics for multiobjective combinatorial optimization problems : Review and recent issues. Research Report RR-5978, INRIA.

BASSEUR, M. et ZITZLER, E. (2006). Handling uncertainty in indicator-based multiobjective optimization. *International Journal of Computational Intelligence Research*, 2(3):255–272.

BEAN, J. (1994). Genetic algorithms and random keys for sequencing and optimization. *ORSA Journal on Computing*, 6(2):154–160.

BEASLEY, J. E. et NASCIMENTO, E. M. (1996). The vehicle routing-allocation problem : A unifying framework. *Top*, 4(1):65–86.

BEUME, N., NAUJOKS, B. et EMMERICH, M. (2007). SMS-EMOA : Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653–1669.

BILLAUT, J.-C., MOUKRIM, A. et SANLAVILLE, E., éditeurs (2005). *Flexibilité et robustesse en ordonnancement*. Hermès Science, Paris, France.

BLEULER, S., LAUMANNS, M., THIELE, L. et ZITZLER, E. (2003). PISA — a platform and programming language independent interface for search algorithms. *In Second International Conference on Evolutionary Multi-Criterion Optimization (EMO 2003)*, volume 2632 de *Lecture Notes in Computer Science*, pages 494–508, Faro, Portugal. Springer-Verlag.

BOISSON, J.-C., JOURDAN, L., TALBI, E.-G. et HORVATH, D. (2008). Parallel multi-objective algorithms for the molecular docking problem. *In IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB 2008)*, Sun Valley Resort, Idaho, USA.

BOISSON, J.-C., MESMOUDI, S., JOURDAN, L. et TALBI., E.-G. (2009). ParadisEO-MO : une plateforme pour le développement de métaheuristiques à base de solution unique. *In Dixième congrès de la Société Française de Recherche Opérationnelle et d'Aide à la décision (ROADEF 2009)*, pages 208–209, Nancy, France.

BRANKE, J., DEB, K., MIETTINEN, K. et SLOWINSKI, R., éditeurs (2008). *Multiobjective Optimization – Interactive and Evolutionary Approaches*, volume 5252 de *Lecture Notes in Computer Science*. Springer-Verlag.

BÜCHE, D., STOLL, P., DORNBERGER, R. et KOUMOUTSAKOS, P. (2002). Multi-objective evolutionary algorithm for the optimization of noisy combustion processes. *IEEE Transaction on Systems, Man, and Cybernetics, part C : Applications and Reviews*, 32(4):2002.

BUI, L. T., ABBASS, H. A. et ESSAM, D. (2005). Fitness inheritance for noisy evolutionary multi-objective optimization. *In Genetic and Evolutionary Computation Conference (GECCO 2005)*, pages 779–785, New York, NY, USA. ACM.

BURKE, E. K. et KENDALL, G., éditeurs (2005). *Search Methodologies : Introductory Tutorials in Optimization and Decision Support Techniques*. Springer-Verlag, Berlin, Germany.

CABALLERO, R., CERDÁ, E., DEL MAR MUÑOZ, M. et REY, L. (2004). Stochastic approach versus multiobjective approach for obtaining efficient solutions in stochastic multiobjective programming problems. *European Journal of Operational Research*, 158(3):633–648.

CAHON, S. (2005). *ParadisEO : une plate-forme pour la conception et le déploiement de méta-heuristiques parallèles hybrides sur grappes et grilles de calcul*. Thèse de doctorat, Université des Sciences et Technologies de Lille, France.

CAHON, S., MELAB, N. et TALBI, E.-G. (2004). ParadisEO : A framework for the reusable design of parallel and distributed metaheuristics. *Journal of Heuristics*, 10(3):357–380.

COELLO COELLO, C. A., AGUIRRE, A. H. et ZITZLER, E., éditeurs (2005). *Third International Conference on Evolutionary Multi-Criterion Optimization (EMO 2005), Proceedings*, volume 3410 de *Lecture Notes in Computer Science*, Guanajuato, Mexico. Springer-Verlag.

COELLO COELLO, C. A., LAMONT, G. B. et VAN VELDHUIZEN, D. A. (2007). *Evolutionary Algorithms for Solving Multi-Objective Problems (second edition)*. Genetic and Evolutionary Computation Series. Springer, New York, USA.

CORNE, D., KNOWLES, J. D. et OATES, M. J. (2000). The pareto envelope-based selection algorithm for multi-objective optimisation. *In Conference on Parallel Problem Solving from Nature (PPSN VI)*, volume 1917 de *Lecture Notes in Computer Science*, pages 839–848, London, UK. Springer-Verlag.

CUNNINGHAM, A. A. et DUTTA, S. K. (1973). Scheduling jobs with exponentially distributed processing times on two machines of a flow shop. *Naval Research Logistics Quarterly*, 16:69–81.

CURRENT, J. R. et SCHILLING, D. A. (1994). The median tour and maximal covering tour problems : Formulations and heuristics. *European Journal of Operational Research*, 73:114–126.

DAUZÈRE-PÉRÈS, S., CASTAGLIOLA, P. et LAHLOU, C. (2005). Niveau de service en ordonnancement stochastique. *In* BILLAUT, J.-C., MOUKRIM, A. et SANLAVILLE, E., éditeurs : *Flexibilité et robustesse en ordonnancement*, chapitre 5, pages 97–113. Hermès Science, Paris, France.

DE JONG, K. A. (1975). *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. Thèse de doctorat, Ann Arbor, University of Michigan.

DEB, K. (2001). *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, Chichester, UK.

DEB, K. (2008). A robust evolutionary framework for multi-objective optimization. *In Genetic and Evolutionary Computation Conference (GECCO 2008)*, pages 633–640, Atlanta, GA, USA. ACM.

DEB, K., AGRAWAL, S., PRATAP, A. et MEYARIVAN, T. (2002). A fast and elitist multiobjective genetic algorithm : NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.

DEB, K., CHAUDHURI, S. et MIETTINEN, K. (2006). Towards estimating nadir objective vector using evolutionary approaches. *In Proceedings of the 8th Genetic and Evolutionary Computation Conference (GECCO 2006)*, pages 643–650, Seattle, Washington, USA.

DEB, K. et GUPTA, H. (2006). Introducing robustness in multi-objective optimization. *Evolutionary Computation*, 14(4):463–494.

DEB, K., MOHAN, M. et MISHRA, S. (2005a). Evaluating the ϵ-domination based multi-objective evolutionary algorithm for a quick computation of pareto-optimal solutions. *Evolutionary Computation*, 13(4):501–525.

DEB, K. et SUNDAR, J. (2006). Reference point based multi-objective optimization using evolutionary algorithms. *In Genetic and Evolutionary Computation Conference (GECCO 2006)*, pages 635–642, Seattle, Washington, USA,. ACM.

DEB, K., THIELE, L., LAUMANNS, M. et ZITZLER, E. (2005b). Scalable test problems for evolutionary multi-objective optimization. *In* ABRAHAM, A., JAIN, R. et GOLDBERG, R., éditeurs : *Evolutionary Multiobjective Optimization : Theoretical Advances and Applications*, chapitre 6, pages 105–145. Springer.

DHAENENS, C., LEMESRE, J. et TALBI, E.-G. (2010). K-PPM : A new exact method to solve multi-objective combinatorial optimization problems. *European Journal of Operational Research*, 200(1):45–53.

DIAS, T. C. S., DE SOUSA FILHO, G. F., MACAMBIRA, E. M., CABRAL, L. A. F. et FAMPA, M. H. C. (2006). An efficient heuristic for the ring star problem. *In Fifth International Workshop on Experimental Algorithms (WEA 2006)*, volume 4007 de *Lecture Notes in Computer Science*, pages 24–35, Menorca, Spain. Springer-Verlag.

DOERNER, K., FOCKE, A. et GUTJAHR, W. J. (2007). Multicriteria tour planning for mobile healthcare facilities in a developing country. *European Journal of Operational Research*, 179(3): 1078–1096.

DURILLO, J. J., NEBRO, A. J., LUNA, F., DORROSORO, B. et ALBA, E. (2006). jMetal : A java framework for developing multi-objective optimization metaheuristics. Rapport technique ITI-2006-10, University of Málaga.

EDGEWORTH, F. Y. (1881). *Mathematical physics*. P. Keagan, London, England.

EHRGOTT, M. (2005). *Multicriteria optimization (2nd edition)*. Springer.

EHRGOTT, M., FONSECA, C. M., GANDIBLEUX, X., HAO, J.-K. et SEVAUX, M., éditeurs (2009). *Fifth International Conference on Evolutionary Multi-Criterion Optimization (EMO 2007), Proceedings*, volume 5467 de *Lecture Notes in Computer Science*, Nantes, France. Springer-Verlag.

EHRGOTT, M. et GANDIBLEUX, X. (2008). Hybrid metaheuristics for multi-objective combinatorial optimization. *In Hybrid Metaheuristics : An Emerging Approach to Optimization*, volume 114 de *Studies in Computational Intelligence*, chapitre 8, pages 221–259. Springer-Verlag, Berlin, Germany.

EHRGOTT, M. et KLAMROTH, K. (1997). Connectedness of efficient solutions in multiple criteria combinatorial optimization. *European Journal of Operational Research*, 97(1):159–166.

EIBEN, A. E. et SMITH, J. E. (2003). *Introduction to Evolutionary Computing*. Springer, New York, USA.

EMMERICH, M., BEUME, N. et NAUJOKS, B. (2005). An EMO algorithm using the hypervolume measure as selection criterion. *In Third International Conference on Evolutionary Multi-criterion Optimization (EMO 2005)*, volume 3410 de *Lecture Notes in Computer Science*, pages 62–76, Guanajuato, Mexico. Springer-Verlag.

ESKANDARI, H. et GEIGER, C. (2009). Evolutionary multiobjective optimization in noisy problem environments. *Journal of Heuristics*. (to appear).

FIELDSEND, J. E., EVERSON, R. et SINGH, S. (2003). Using unconstrained elite archives for multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 7(3):305–323.

FIELDSEND, J. E. et EVERSON, R. M. (2005). Multi-objective optimisation in the presence of uncertainty. *In IEEE Congress on Evolutionary Computation (CEC 2005)*, pages 243–250, Edinburgh, UK. IEEE Press.

FIGUEIRA, J. R., LIEFOOGHE, A., TALBI, E.-G. et WIERZBICKI, A. P. (2009). A parallel multiple reference point approach for multi-objective optimization. *European Journal of Operational Research*. (à paraître).

FONSECA, C. M. et FLEMING, P. J. (1993). Genetic algorithms for multiobjective optimization : Formulation, discussion and generalization. *In 5th International Conference on Genetic Algorithms (ICGA 1993)*, pages 416–423, Urbana-Champaign, IL, USA. Morgan Kaufmann.

FONSECA, C. M., FLEMING, P. J., ZITZLER, E., DEB, K. et THIELE, L., éditeurs (2003). *Second International Conference on Evolutionary Multi-Criterion Optimization (EMO 2003), Proceedings*, volume 2632 de *Lecture Notes in Computer Science*, Faro, Portugal. Springer-Verlag.

FORTEMPS, P. (1997). Jobshop scheduling with imprecise durations : A fuzzy approach. *IEEE Transactions On Fuzzy System*, 5(4):557–569.

FOURMAN, M. P. (1985). Compaction of symbolic layout using genetic algorithms. *In 1st International Conference on Genetic Algorithms (ICGA 1985)*, pages 141–153, Pittsburgh, PA, USA. Lawrence Erlbaum Associates.

GAGNÉ, C. et PARIZEAU, M. (2006). Genericity in evolutionary computation software tools : Principles and case study. *International Journal on Artificial Intelligence Tools*, 15(2):173–194.

GANDIBLEUX, X., MEZDAOUI, N. et FRÉVILLE, A. (1997). A tabu search procedure to solve multiobjective combinatorial optimization problems. *In Advances in Multiple Objective and Goal Programming*, volume 455 de *Lecture Notes in Economics and Mathematical Systems*, pages 291–300. Springer-Verlag, Berlin, Germany.

GASPAR-CUNHA, A. et COVAS, J. A. (2008). Robustness in multi-objective optimization using evolutionary algorithms. *Computational Optimization and Applications*, 39(1):75–96.

GENDREAU, M., HERTZ, A. et LAPORTE, G. (1992). New insertion and postoptimization procedures for the traveling salesman problem. *Operations Research*, 40(6):1086–1094.

GOH, C. K. et TAN, K. C. (2007a). Evolving the tradeoffs between Pareto-optimality and robustness in multi-objective evolutionary algorithms. *In* YANG, S., ONG, Y.-S. et JIN, Y., éditeurs : *Evolutionary Computation in Dynamic and Uncertain Environments*, volume 51 de *Studies in Computational Intelligence*, chapitre 20, pages 457–478. Springer-Verlag.

GOH, C. K. et TAN, K. C. (2007b). An investigation on noisy environments in evolutionary multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 11(3):354–381.

GOH, C. K., TAN, K. C., CHEONG, C. Y. et ONG, Y. S. (2007). Noise-induced features in robust multi-objective optimization problems. *In IEEE Congress on Evolutionary Computation (CEC 2007)*, pages 568–575, Singapore. IEEE Press.

GOLDBERG, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, Boston, MA, USA.

GOLDBERG, D. E. et RICHARDSON, J. (1987). Genetic algorithms with sharing for multimodal function optimization. *In 2nd International Conference on Genetic Algorithms and their application*, pages 41–49, Mahwah, NJ, USA. Lawrence Erlbaum Associates, Inc.

GOURGAND, M., GRANGEON, N. et NORRE, S. (2005). Modèle du flow-shop de permutation stochastique. *In* BILLAUT, J.-C., MOUKRIM, A. et SANLAVILLE, E., éditeurs : *Flexibilité et robustesse en ordonnancement*, chapitre 7, pages 135–161. Hermès Science, Paris, France.

GRAHAM, R. L., LAWLER, E. L., LENSTRA, J. K. et RINNOOY KAN, A. H. G. (1979). Optimization and approximation in deterministic sequencing and scheduling : A survey. *Annals of Discrete Mathematics*, 5:287–326.

HANSEN, M. P. (1997). Tabu search for multiobjective optimisation : MOTS. *In Thirteenth International Conference on Multiple Criteria Decision Making (MCDM 1997)*, Cape Town, South Africa. Springer-Verlag.

HANSEN, M. P. et JASZKIEWICZ, A. (1998). Evaluating the quality of approximations of the nondominated set. Rapport technique IMM-REP-1998-7, Institute of Mathematical Modeling, Technical University of Denmark.

HANSEN, P. (1979). Bicriterion path problems. *In Multiple Criteria Decision Making : Theory and Applications*, volume 177 de *Lecture Notes in Economics and Mathematical Systems*, pages 109–127, Berlin, Germany. Springer-Verlag.

HELBIG, S. et PATEVA, D. (1994). On several concepts for ϵ-efficiency. *OR Spektrum*, 16(3):179–186.

HOLLAND, J. H. (1975). *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Abor, MI, USA.

HORN, J., NAFPLIOTIS, N. et GOLDBERG, D. E. (1994). A niched pareto genetic algorithm for multiobjective optimization. *In IEEE Congress on Evolutionary Computation (CEC 1994)*, pages 82–87, Piscataway, NJ, USA. IEEE Press.

HUGHES, E. J. (2001). Evolutionary multi-objective ranking with uncertainty and noise. *In First International Conference on Evolutionary Multi-Criterion Optimization (EMO 2001)*, volume 1993 de *Lecture Notes in Computer Science*, pages 329–343, London, UK. Springer-Verlag.

IGEL, C., GLASMACHERS, T. et HEIDRICH-MEISNER, V. (2008). Shark. *Journal of Machine Learning Research*, 9:993–996.

ISHIBUCHI, H. et MURATA, T. (1998). A multi-objective genetic local search algorithm and its application to flowshop scheduling. *IEEE Transactions on Systems, Man, and Cybernetics, Part C : Applications and Reviews*, 28(3):392–403.

ISHIBUCHI, H., YOSHIDA, T. et MURATA, T. (2003). Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. *IEEE Transactions on Evolutionary Computation*, 7(2):204–223.

JASZKIEWICZ, A. (2002). Genetic local search for multi-objective combinatorial optimization. *European Journal of Operational Research*, 137(1):50–71.

JIN, Y. et BRANKE, J. (2005). Evolutionary optimization in uncertain environments - a survey. *IEEE Transactions on Evolutionary Computation*, 9:303–317.

JIN, Y. et SENDHOFF, B. (2003). Trade-off between optimality and robustness : An evolutionary multiobjective approach. *In* FONSECA, C. M., FLEMING, P. J., ZITZLER, E., DEB, K. et THIELE, L., éditeurs : *Second International Conference on Evolutionary Multi-Criterion Optimization (EMO 2003)*, volume 2632 de *Lecture Notes in Computer Science*, pages 237–251, Faro, Portugal. Springer-Verlag.

JOZEFOWIEZ, N. (2004). *Modélisation et résolution approchée de problèmes de tournées de véhicules*. Thèse de doctorat, Université des Sciences et Technologies de Lille, France.

JOZEFOWIEZ, N., SEMET, F. et TALBI, E.-G. (2008). Multi-objective vehicle routing problems. *European Journal of Operational Research*, 189(2):293–309.

KEIJZER, M., MERELO, J.-J., ROMERO, G. et SCHOENAUER, M. (2001). Evolving objects : A general purpose evolutionary computation library. *In 5th International Conference on Artificial Evolution (EA 2001)*, pages 231–244, Le Creusot, France.

KNOWLES, J. et CORNE, D. (2004). Bounded Pareto archiving : Theory and practice. *In Metaheuristics for Multiobjective Optimisation*, volume 535 de *Lecture Notes in Economics and Mathematical Systems*, chapitre 2, pages 39–64. Springer-Verlag, Berlin, Germany.

KNOWLES, J. et CORNE, D. (2005). Memetic algorithms for multiobjective optimization : Issues, methods and prospects. *In Recent Advances in Memetic Algorithms*, volume 166 de *Studies in Fuzziness and Soft Computing*, pages 313–352. Springer.

KNOWLES, J. D. et CORNE, D. (2000). Approximating the nondominated front using the Pareto archived evolution strategy. *Evolutionary Computation*, 8(2):149–172.

KOUVELIS, P., DANIELS, R. L. et VAIRAKTARAKIS, G. (2000). Robust scheduling of a two-machine flow shop with uncertain processing times. *IIE Transactions*, 32(5):421–432.

KOUVELIS, P. et YU, G. (1997). *Robust discrete optimization and its applications.* Kluwer Academic.

KU, P. S. et NIU, S. C. (1986). On johnson's two-machine flow shop with random processing times. *Operations Research*, 34:130–136.

LABBÉ, M. et LAPORTE, G. (1986). Maximizing user convenience and postal service effiency in post box location. *Belgian Journal of Operations Research, Statistics and Computer Science*, 26:21–35.

LABBÉ, M., LAPORTE, G. et RODRÍGUEZ MARTÍN, I. (1998). Path, tree and cycle location. *In* CRAINIC, T. et LAPORTE, G., éditeurs : *Fleet Management and Logistics*, pages 187–204. Kluwer Academic Publishers, Boston, MA, USA.

LABBÉ, M., LAPORTE, G., RODRÍGUEZ MARTÍN, I. et SALAZAR GONZÁLEZ, J. J. (1999). The median cycle problem. Working paper, CRT-99-29, Université de Montréal.

LABBÉ, M., LAPORTE, G., RODRÍGUEZ MARTÍN, I. et SALAZAR GONZÁLEZ, J. J. (2004). The ring star problem : Polyhedral analysis and exact algorithm. *Networks*, 43:177–189.

LABBÉ, M., LAPORTE, G., RODRÍGUEZ MARTÍN, I. et SALAZAR GONZÁLEZ, J. J. (2005). Locating median cycles in networks. *European Journal of Operational Research*, 160(2):457–470.

LANDA SILVA, J. D., BURKE, E. et PETROVIC, S. (2004). An introduction to multiobjective metaheuristics for scheduling and timetabling. *In Metaheuristics for Multiobjective Optimisation*, volume 535 de *Lecture Notes in Economics and Mathematical Systems*, pages 91–129. Springer-Verlag, Berlin, Germany.

LAUMANNS, M., THIELE, L., DEB, K. et ZITZLER, E. (2002). Combining convergence and diversity in evolutionary multi-objective optimization. *Evolutionary Computation*, 10(3):263–282.

LAUMANNS, M., ZITZLER, E. et THIELE, L. (2000). A unified model for multi-objective evolutionary algorithms with elitism. *In IEEE Congress on Evolutionary Computation (CEC 2000)*, pages 46–53, Piscataway, New Jersey, USA. IEEE Press.

LAUMANNS, M., ZITZLER, E. et THIELE, L. (2001). On the effects of archiving, elitism, and density based selection in evolutionary multi-objective optimization. *In* ZITZLER, E., DEB, K., THIELE, L., COELLO COELLO, C. A. et CORNE, D., éditeurs : *International Conference on Evolutionary Multi-Criterion Optimization (EMO 2001)*, volume 1993 de *Lecture Notes in Computer Science*, pages 181–196, Zurich, Switzerland. Springer-Verlag.

LEMESRE, J. (2006). *Méthodes exactes pour l'optimisation combinatoire multi-objectif : conception et application.* Thèse de doctorat, Université des Sciences et Technologies de Lille, France.

LEMESRE, J., DHAENENS, C. et TALBI, E. (2007a). An exact parallel method for a bi-objective permutation flowshop problem. *European Journal of Operational Research*, 177(3):1641 – 1655.

LEMESRE, J., DHAENENS, C. et TALBI, E. (2007b). Parallel partitioning method (PPM) : A new exact method to solve bi-objective problems. *Computers & Operations Research*, 34(8):2450–2462.

LIEFOOGHE, A., BASSEUR, M., JOURDAN, L. et TALBI, E.-G. (2006). Optimisation multi-objectif sous incertitude pour le flow-shop de permutation. *In META 2006*, Hammamet, Tunisia.

LIEFOOGHE, A., BASSEUR, M., JOURDAN, L. et TALBI, E.-G. (2007a). Combinatorial optimization of stochastic multi-objective problems : an application to the flow-shop scheduling problem. *In Fourth International Conference on Evolutionary Multi-criterion Optimization (EMO 2007)*, volume 4403 de *Lecture Notes in Computer Science*, pages 457–471, Matsushima, Japan. Springer-Verlag.

LIEFOOGHE, A., BASSEUR, M., JOURDAN, L. et TALBI, E.-G. (2007b). ParadisEO-MOEO : A framework for evolutionary multi-objective optimization. *In Fourth International Conference on Evolutionary Multi-criterion Optimization (EMO 2007)*, volume 4403 de *Lecture Notes in Computer Science*, pages 386–400, Matsushima, Japan. Springer-Verlag.

LIEFOOGHE, A., JOURDAN, L., BASSEUR, M. et TALBI, E.-G. (2008a). Métaheuristiques pour le flow-shop de permutation bi-objectif stochastique. *Revue d'Intelligence Artificielle (RIA)*, 22(2):183–208.

LIEFOOGHE, A., JOURDAN, L., BASSEUR, M., TALBI, E.-G. et BURKE, E. K. (2008b). Metaheuristics for the bi-objective ring star problem. *In Eighth European Conference on Evolutionary Computation in Combinatorial Optimisation (EvoCOP 2008)*, volume 4472 de *Lecture Notes in Computer Science*, pages 206–217, Napoli, Italy. Springer-Verlag.

LIEFOOGHE, A., JOURDAN, L., JOZEFOWIEZ, N. et TALBI, E.-G. (2008c). On the integration of a TSP heuristic into an EA for the bi-objective ring star problem. *In International Workshop on Hybrid Metaheuristics (HM 2008)*, volume 5296 de *Lecture Notes in Computer Science*, pages 117–130, Malaga, Spain. Springer-Verlag.

LIEFOOGHE, A., JOURDAN, L., LEGRAND, T., HUMEAU, J. et TALBI, E.-G. (2009a). ParadisEO-MOEO : A software framework for evolutionary multi-objective optimization. *In Advances in multi-objective nature inspired computing*, Studies in Computational Intelligence. Springer-Verlag. (à paraître).

LIEFOOGHE, A., JOURDAN, L. et TALBI, E.-G. (2007c). Software framework for multi-objective optimization. *In 22nd European Conference on Operational Research (EURO XXII)*, Prague, Czech Republic.

LIEFOOGHE, A., JOURDAN, L. et TALBI, E.-G. (2008d). Metaheuristics and hybrid metaheuristics for the bi-objective ring star problem. *In International Conference on Metaheuristics and Nature Inspired Computing (META 2008)*, Hammamet, Tunisia.

LIEFOOGHE, A., JOURDAN, L. et TALBI, E.-G. (2008e). Metaheuristics and their hybridization to solve the bi-objective ring star problem : a comparative study. Research Report 6515, INRIA.

LIEFOOGHE, A., JOURDAN, L. et TALBI, E.-G. (2009b). Evolutionary multiobjective optimization in uncertain environments : New approaches, performance assessment and a comparative study on flowshop scheduling. *IEEE Transactions on Evolutionary Computation*. (soumis).

LIEFOOGHE, A., JOURDAN, L. et TALBI, E.-G. (2009c). The ParadisEO-MOEO software framework : A unified design and implementation of evolutionary multiobjective optimization algorithms. *European Journal of Operational Research*. (soumis).

LIEFOOGHE, A., JOURDAN, L. et TALBI, E.-G. (2009d). A unified model for evolutionary multi-objective optimization and its implementation in a general purpose software framework. *In IEEE Symposium on Computational Intelligence in Multicriteria Decision-Making (IEEE MCDM 2009)*, pages 88–95, Nashville, Tennessee, USA. IEEE Press.

LIEFOOGHE, A., JOURDAN, L. et TALBI, E.-G. (2010). Metaheuristics and cooperative approaches for the bi-objective ring star problem. *Computers & Operations Research*, 37(6):1033–1044.

LIEFOOGHE, A., MESMOUDI, S., HUMEAU, J., JOURDAN, L. et TALBI, E.-G. (2009e). On dominance-based local search : Design, implementation and experiments on multiobjective scheduling and traveling salesman problems. *Journal of Heuristics*. (soumis).

LIEFOOGHE, A., MESMOUDI, S., HUMEAU, J., JOURDAN, L. et TALBI, E.-G. (2009f). A study on dominance-based local search approaches for multiobjective combinatorial optimization. *In Workshop on Engineering Stochastic Local Search Algorithms. Designing, Implementing and Analyzing Effective Heuristics (SLS 2009)*, volume 5752 de *Lecture Notes in Computer Science*, pages 120–124, Brussels, Belgium. Springer-Verlag.

LIMBOURG, P. (2005). Multi-objective optimization of problems with epistemic uncertainty. *In* COELLO, C. A. C., AGUIRRE, A. H. et ZITZLER, E., éditeurs : *Third International Conference on Evolutionary Multi-Criterion Optimization (EMO 2005)*, volume 3410 de *Lecture Notes in Computer Science*, pages 413–427, Guanajuato, Mexico. Spriner-Verlag.

LIMBOURG, P. et SALAZAR APONTE, D. E. (2005). An optimization algorithm for imprecise multi-objective problem functions. *In IEEE Congress on Evolutionary Computation (CEC 2005)*, pages 459–466, Edinburgh, UK. IEEE Press.

LOURENCO, H. R., MARTIN, O. et STÜTZLE, T. (2002). Iterated local search. *In* GLOVER, F. et KOCHENBERGER, G., éditeurs : *Handbook of Metaheuristics*, volume 57 de *International Series in Operations Research & Management Science*, chapitre 11, pages 321–353. Kluwer Academic Publishers, Norwell, MA, USA.

LUST, T. et TEGHEM, J. (2009). Two-phase pareto local search for the biobjective traveling salesman problem. *Journal of Heuristics*. (to appear).

MADDOX, M. J. et BIRGE, J. R. (1996). Using second moment information in stochastic scheduling. *In Recent advances in control and optimization of manufacturing systems*, volume 214 de *Lecture Notes in Control and Information Sciences*, pages 99–120, London, UK. Springer.

MAKINO, T. (1965). On a scheduling problem. *Journal of the Operations Research Society of Japan*, 8:32–44.

MAUTTONE, A., NESMACHNOW, S., OLIVERA, A. et ROBLEDO, F. (2007). A hybrid metaheuristic algorithm to solve the capacitated m-ring star problem. *In International Network Optimization Conference (INOC 2007)*, Spa, Belgium.

MELAB, N., TALBI, E.-G., CAHON, S., ALBA, E. et LUQUE, G. (2006). Parallel metaheuristics : Models and frameworks. *In* TALBI, E.-G., éditeur : *Parallel Combinatorial Optimization*, chapitre 6, pages 149–162. John Wiley & Sons, Chichester, UK.

MEUNIER, H., TALBI, E.-G. et REININGER, P. (2000). A multiobjective genetic algorithm for radio network optimization. *In IEEE Congress on Evolutionary Computation (CEC 2000)*, pages 317–324, San Diego, USA. IEEE Press.

MIETTINEN, K. (1999). *Nonlinear Multiobjective Optimization*, volume 12 de *International Series in Operations Research and Management Science*. Kluwer Academic Publishers, Boston, MA, USA.

MINELLA, G., RUIZ, R. et CIAVOTTA, M. (2008). A review and evaluation of multi-objective algorithms for the flowshop scheduling problem. *INFORMS Journal on Computing*, 20(3):451–471.

MOLINA, J., SANTANA, L. V., HERNÁNDEZ-DÍAZ, A. G., COELLO COELLO, C. A. et CABALLERO, R. (2009). g-dominance : Reference point based dominance for multiobjective metaheuristics. *European Journal of Operational Research*, 167(2):685–692.

MORENO PÉREZ, J. A., MORENO-VEGA, J. M. et RODRÍGUEZ MARTÍN, I. (2003). Variable neighborhood tabu search and its application to the median cycle problem. *European Journal of Operations Research*, 151(2):365–378.

MOSTAGHIM, S., TEICH, J. et TYAGI, A. (2002). Comparison of data structures for storing pareto-sets in MOEAs. *In IEEE Congress on Evolutionary Computation (CEC 2002)*, pages 843–848, Piscataway, NJ, USA. IEEE Press.

NAGAR, A., HADDOCK, J. et HERAGU, S. (1995). Multiple and bicriteria scheduling : A literature survey. *European Journal of Operational Research*, 81:88–104.

OBAYASHI, S., DEB, K., POLONI, C., HIROYASU, T. et MURATA, T., éditeurs (2007). *Fourth International Conference on Evolutionary Multi-Criterion Optimization (EMO 2007), Proceedings*, volume 4403 de *Lecture Notes in Computer Science*, Matsushima, Japan. Springer-Verlag.

PAQUETE, L., CHIARANDINI, M. et STÜTZLE, T. (2004). Pareto local optimum sets in the biobjective traveling salesman problem : An experimental study. *In* GANDIBLEUX, X., SEVAUX, M., SÖRENSEN, K. et T'KINDT, V., éditeurs : *Metaheuristics for Multiobjective Optimisation*, volume 535 de *Lecture Notes in Economics and Mathematical Systems*, chapitre 7, pages 177–199. Springer-Verlag, Berlin, Germany.

PAQUETE, L. et STÜTZLE, T. (2003). A two-phase local search for the biobjective traveling salesman problem. *In Second International Conference on Evolutionary Multi-Criterion Optimization (EMO 2003)*, volume 2632 de *Lecture Notes in Computer Science*, pages 479–493, Faro, Portugal. Springer-Verlag.

PAQUETE, L. et STÜTZLE, T. (2009). Design and analysis of stochastic local search for the multiobjective traveling salesman problem. *Computers & Operations Research*, 36(9):2619–2631.

PARETO, V. (1896). *Cours d'économie politique*, volume 1–2. Rouge, Lausanne, Switzerland.

POLES, S., VASSILEVA, M. et SASAKI, D. (2008). Multiobjective optimization software. *In* BRANKE, J., DEB, K., MIETTINEN, K. et SLOWÍNSKI, R., éditeurs : *Multiobjective Optimization - Interactive and Evolutionary Approaches*, volume 5252 de *Lecture Notes in Computer Science (LNCS)*, chapitre 12, pages 329–348. Springer-Verlag, Berlin Heidelberg.

RAY, T. (2002). Constrained robust optimal design using a multiobjective evolutionary algorithm. *In IEEE Congress on Evolutionary Computation (CEC 2002)*, pages 419–424, Honolulu, HI, USA. IEEE Press.

REINELT, G. (1991). TSPLIB – A traveling salesman problem library. *ORSA Journal on Computing*, 3(4):376–384.

RENAUD, J., BOCTOR, F. F. et LAPORTE, G. (2004). Efficient heuristics for median cycle problems. *Journal of the Operational Research Society*, 55(2):179–186.

ROY, B. (2005). À propos de robustesse en recherche opérationnelle et aide à la décision. *In* BILLAUT, J.-C., MOUKRIM, A. et SANLAVILLE, E., éditeurs : *Flexibilité et robustesse en ordonnancement*, chapitre 2, pages 35–50. Hermès Science, Paris, France.

RUDOLPH, G. et AGAPIE, A. (2000). Convergence properties of some multi-objective evolutionary algorithms. *In IEEE Congress on Evolutionary Computation (CEC 2000)*, pages 1010–1016, Piscataway, New Jersey, USA. IEEE Press.

SCHAFFER, J. D. (1985). Multiple objective optimization with vector evaluated genetic algorithms. *In 1st International Conference on Genetic Algorithms (ICGA 1985)*, pages 93–100, Pittsburgh, PA, USA. Lawrence Erlbaum Associates.

SCHIAVINOTTO, T. et STÜTZLE, T. (2007). A review of metrics on permutations for search landscape analysis. *Computers & Operations Research*, 34(10):3143–3153.

SCHOTT, J. R. (1995). Fault tolerant design using single and multicriteria genetic algorithm optimization. Mémoire de D.E.A., Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, Massachusetts, Cambridge, Massachusetts, USA.

SCHUETZE, O., JOURDAN, L., LEGRAND, T., TALBI, E.-G. et WOJKIEWICZ, J.-L. (2008). New analysis of the optimization of electromagnetic shielding properties using conducting polymers and a multi-objective approach. *Polymers for Advanced Technologies*, 19(7):762–769.

SERAFINI, P. (1992). Simulated annealing for multiobjective optimization problems. *In Tenthth International Conference on Multiple Criteria Decision Making (MCDM 1992)*, pages 87–96, Taipei, Taiwan.

SILVERMAN, B. (1986). *Density Estimation for Statistics and Data Analysis*. Monographs on Statistics and Applied Probability. Chapman & Hall, London, UK.

SINGH, A. (2003). Uncertainty based multi-objective optimization of groundwater remediation design. Mémoire de D.E.A., University of Illinois at Urbana-Champaign.

SLOWINSKI, R. et TEGHEM, J., éditeurs (1990). *Stochastic versus Fuzzy approaches to Multiobjective Mathematical Programming under uncertainty*. Kluwer Academic, Dordrecht.

SÖRENSEN, K. (2003). *A framework for robust and flexible optimisation using metaheuristics, with applications in supply chain design*. Thèse de doctorat, University of Antwerp, Belgium.

SRINIVAS, N. et DEB, K. (1994). Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248.

STEUER, R. E. (1986). *Multiple Criteria Optimization : Theory, Computation and Application*. John Wiley & Sons, Chichester, UK.

STREICHERT, F. et ULMER, H. (2005). JavaEvA : a java based framework for evolutionary algorithms. Rapport technique WSI-2005-06, Centre for Bioinformatics Tübingen (ZBIT) of the Eberhard-Karls-University, Tübingen.

SZCZEPAŃSKI, M. et WIERZBICKI, A. (2003). Application of multiple criteria evolutionary algorithms to vector optimisation, decision support and reference point approaches. *Journal of Telecommunications and Information Technology*, 3:16–33.

TAILLARD, E. D. (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research*, 64:278–285.

TALBI, E.-G. (2002). A taxonomy of hybrid metaheuristics. *Journal of Heuristics*, 8(2):541–564.

TALBI, E.-G. (2009). *Metaheuristics : from design to implementation*. Wiley.

TALBI, E.-G., CAHON, S. et MELAB, N. (2007). Designing cellular networks using a parallel hybrid metaheuristic on the computational grid. *Computer Communications*, 30(4):698–713.

TALBI, E.-G., JOURDAN, L., GARCIA-NIETO, J. et ALBA, E. (2008). Comparison of population based metaheuristics for feature selection : Application to microarray data classification. *In IEEE/ACS International Conference on Computer Systems and Applications (AICCSA 2008)*, pages 45–52. IEEE Press.

TALBI, E.-G., LIEFOOGHE, A. et JOURDAN, L. (2009a). Multi-objective metaheuristics : from design to implementation. *In 23rd European Conference on Operational Research (EURO XXIII)*, Bonn, Germany.

TALBI, E.-G., LIEFOOGHE, A. et JOURDAN, L. (2009b). A unifed view of EMO : from design to implementation. *In 20th International Conference on Multiple Criteria Decision Making (MCDM 2009)*, Chengdu - Jiuzhaigou, China.

TALBI, E.-G. et MEUNIER, H. (2006). Hierarchical parallel approach for GSM mobile network design. *Journal of Parallel and Distributed Computing*, 66(2):274–290.

TALBI, E.-G., RAHOUAL, M., MABED, M. H. et DHAENENS, C. (2001). A hybrid evolutionary approach for multicriteria optimization problems : Application to the flow shop. *In First International Conference on Evolutionary Multi-criterion Optimization (EMO 2001)*, volume 1993 de *Lecture Notes in Computer Science*, pages 416–428, Zurich, Switzerland. Springer-Verlag.

TALWAR, P. P. (1967). A note on sequencing problems with uncertain job times. *Journal of the Operations Research Society of Japan*, 9:93–97.

TAN, K. C., CHEONG, C. Y. et GOH, C. K. (2007). Solving multiobjective vehicle routing problem with stochastic demand via evolutionary computation. *European Journal of Operational Research*, 177(2):813–839.

TAN, K. C. et GOH, C. K. (2008). Handling uncertainties in evolutionary multi-objective optimization. *In* ZURADA, J. M., YEN, G. G. et WANG, J., éditeurs : *IEEE World Congress on Computational Intelligence (WCCI 2008)*, volume 5050 de *Lecture Notes in Computer Science*, pages 262–292, Hong Kong, China. Springer.

TAN, K. C., LEE, T. H., KHOO, D. et KHOR, E. F. (2001). A multi-objective evolutionary algorithm toolbox for computer-aided multi-objective optimization. *IEEE Transactions on Systems, Man and Cybernetics : Part B (Cybernetics)*, 31(4):537–556.

TEICH, J. (2001). Pareto-front exploration with uncertain objectives. *In First International Conference on Evolutionary Multi-criterion Optimization (EMO 2001)*, volume 1993 de *Lecture Notes in Computer Science*, pages 314–328, London, UK. Springer-Verlag.

THIELE, L., MIETTINEN, K., KORHONEN, P. J. et MOLINA, J. (2009). A preference-based evolutionary algorithm for multi-objective optimization. *Evolutionary Computation*, 17(3):411–436.

T'KINDT, V. et BILLAUT, J.-C. (2002). *Multicriteria Scheduling : Theory, Models and Algorithms*. Springer-Verlag, Berlin, Germany.

TRAUTMANN, H., MEHNEN, J. et NAUJOKS, B. (2009). Pareto-dominance in noisy environments. *In IEEE Congress on Evolutionary Computation (CEC 2009)*, pages 3119–3126, Trondheim, Norway. IEEE Press.

ULUNGU, E. L. et TEGHEM, J. (1995). The two phases method : An efficient procedure to solve bi-objective combinatorial optimization problems. *Foundations of Computing and Decision Sciences*, 20(2):149–165.

ULUNGU, E. L., TEGHEM, J., FORTEMPS, P. et TUYTTENS, D. (1999). MOSA method : a tool for solving multiobjective combinatorial optimization problems. *Journal of Multi-Criteria Decision Analysis*, 8(4):221–236.

VALENZUELA, C. L. (2002). A simple evolutionary algorithm for multi-objective optimization (SEAMO). *In IEEE Congress on Evolutionary Computation (CEC 2002)*, pages 717–722, Piscataway, NJ, USA. IEEE Press.

VELDHUIZEN, D. A. V. et LAMONT, G. B. (2000). On measuring multiobjective evolutionary algorithm performance. *In IEEE Congress on Evolutionary Computation (CEC 2000)*, pages 204–211, Piscataway, NJ, USA. IEEE Press.

VOGT, L., POOJARI, C. et BEASLEY, J. (2007). A tabu search algorithm for the single vehicle routing allocation problem. *Journal of the Operational Research Society*, 58(4):467–480.

WAGNER, T., BEUME, N. et NAUJOKS, B. (2007). Pareto-, aggregation-, and indicator-based methods in many-objective optimization. *In Fourth International Conference on Evolutionary Multi-criterion Optimization (EMO 2007)*, volume 4403 de *Lecture Notes in Computer Science*, pages 742–756, Matsushima, Japan. Springer-Verlag.

WANG, L., ZHANG, L. et ZHENG, D.-Z. (2005). A class of hypothesis-test based genetic algorithm for flow shop scheduling with stochastic processing time. *The International Journal of Advanced Manufacturing Technology*, 25(11-12):1157–1163.

WIERZBICKI, A. (1980). The use of reference objectives in multiobjective optimization. *In* FANDEL, G. et GAL, T., éditeurs : *Multiple Objective Decision Making, Theory and Application*, volume 177 de *Lecture Notes in Economics and Mathematical Systems*, pages 468–486. Springer-Verlag.

WIERZBICKI, A. P., MAKOWSKI, M. et WESSELS, J., éditeurs (2000). *Model-Based Decision Support Methodology with Environmental Applications*. Kluwer Academic Publishers, Dordrecht.

XU, J., CHIU, S. Y. et GLOVER, F. (1999). Optimizing a ring-based private line telecommunication network using tabu search. *Management Science*, 45(3):330–345.

ZITZLER, E., DEB, K., THIELE, L., COELLO, C. A. C. et CORNE, D., éditeurs (2001a). *First International Conference on Evolutionary Multi-Criterion Optimization (EMO 2001), Proceedings*, volume 1993 de *Lecture Notes in Computer Science*, Zurich, Switzerland. Springer-Verlag.

ZITZLER, E., KNOWLES, J. et THIELE, L. (2008). Quality assessment of pareto set approximations. *In* BRANKE, J., DEB, K., MIETTINEN, K. et SLOWÍNSKI, R., éditeurs : *Multiobjective Optimization - Interactive and Evolutionary Approaches*, volume 5252 de *Lecture Notes in Computer Science*, chapitre 14, pages 373–404. Springer-Verlag, Berlin, Germany.

ZITZLER, E. et KÜNZLI, S. (2004). Indicator-based selection in multiobjective search. *In Conference on Parallel Problem Solving from Nature (PPSN VIII)*, volume 3242 de *Lecture Notes in Computer Science*, pages 832–842, Birmingham, UK. Springer-Verlag.

ZITZLER, E., LAUMANNS, M. et BLEULER, S. (2004). A tutorial on evolutionary multiobjective optimization. *In Metaheuristics for Multiobjective Optimisation*, volume 535 de *Lecture Notes in Economics and Mathematical Systems*, chapitre 1, pages 3–38. Springer-Verlag, Berlin, Germany.

ZITZLER, E., LAUMANNS, M. et THIELE, L. (2001b). SPEA2 : Improving the strength pareto evolutionary algorithm. TIK Report 103, Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Zurich, Switzerland.

ZITZLER, E. et THIELE, L. (1999). Multiobjective evolutionary algorithms : A comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271.

ZITZLER, E., THIELE, L., LAUMANNS, M., FONESECA, C. M. et GRUNERT DA FONSECA, V. (2003). Performance assessment of multiobjective optimizers : An analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132.